Neural Matching and Importance Learning in Information Retrieval

Zhuyun Dai

CMU-LTI-20-011

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Jamie Callan (Chair), Carnegie Mellon University
Graham Neubig, Carnegie Mellon University
Tie-Yan Liu, Carnegie Mellon University & Microsoft Research
Yiqun Liu, Tsinghua University

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in Language and Information Technologies

To my parents Mr. Yuchun Dai and Mrs. Junxia Dai.

Abstract

For 50-60 years, information retrieval (IR) systems have relied on bag-of-words approaches. Although bag-of-words retrieval has several long-standing limitations, attempts to solve these issues have been mostly unsuccessful. Recently, neural networks provide a new paradigm for modeling natural languages. This dissertation combines insights from IR and the key advantages of neural networks to address two important challenges of IR in language understanding.

The first part of this dissertation focuses on how queries and documents are matched. State-of-the-art rankers have previously relied on exact lexical match, which causes the well-known vocabulary mismatch problem. This dissertation develops neural models that bring *soft match* into relevance ranking. Using distributed text representations, our models can soft match every query word to every document word. As the soft match signals are noisy, this dissertation presents a novel kernel-pooling technique that groups soft matches based on their contribution to relevance. This dissertation also studies whether pre-trained model parameters can improve low-resource domains, and whether the model architectures are re-usable in a non-text retrieval task. Our approaches outperform previous state-of-the-art ranking systems by large margins.

The second part of this dissertation focuses on how queries and documents are represented. A typical search engine uses frequency statistics to weight words, but frequent words are not necessarily essential to the meaning of the text. This dissertation develops neural networks to estimate *word importance* based on how a word interacts with its linguistic context. A weak-supervision approach is developed that allows training our models without any human annotations. Our models can run offline, significantly improving first-stage retrieval without hurting efficiency.

To summarize, this dissertation formulates a new neural retrieval paradigm that overcomes classic retrieval models' limitations in matching and importance weighting. It points out several promising paths in neural relevance ranking, deep retrieval models, and deep document understanding for IR.

Acknowledgments

The journey towards a PhD was challenging in many ways. The completion of my dissertation would not have been possible without the guidance from my advisor and mentors, help from friends, and support from family.

I would like to express my deepest appreciation to my advisor, Jamie Callan. Looking back at my PhD journey, Jamie provided careful guidance when I started, "forced" me to be independent when I should, and gave me plenty freedom when I became ready. When I was struggling, his encouraging words made me feel better about myself. When I felt too good about myself, he wrote "green comments" all over my research papers and this dissertation, challenging me to think deep into the fundamental research problems. I was fortunate to have Jamie as my advisor, for he set a high standard about how to be a researcher and a good person.

I also wish to thank my other thesis committee members Tie-Yan Liu, Graham Neubig, and Yiqun Liu. I was consistently amazed by Tie-Yan's vision and breadth of knowledge. His sharp questions drive me towards more novel and more impactful research. Graham was always approachable and supportive. My chats with Graham often improved my understanding of the topic. I sincerely thank Yiqun for agreeing to serve as the external member of my thesis committee and steering me through this dissertation with insightful feedback.

I am grateful for my collaborators and colleagues. Special thanks to Chenyan Xiong for being a helpful co-author, an excellent mentor, and a reliable friend. I would not have chosen this exciting dissertation research topic without the countless help from Chenyan. Thanks to Yubin Kim for mentoring my first research project, helping me with my first paper, and taking me to my first conference. To Luyu Gao, Zhen Fan, Joel Mackenzie, and Luke Gallagher: this dissertation benefited much from your efforts. Thank you to Shuo Zhang for introducing me to new research areas. Thank you to my officemate and friend, Zi Yang, for always being supportive. Thanks to my colleagues from the Callan Group and friends from CMU. This dissertation research would have been less fun without you.

To my brilliant roommates, Fuchen Liu and Shurui Zhou. I was blessed to have you as my best friends. Thank you for always being by my side throughout the past six years.

To Zhongwen Wang, my high-school teacher. Thank you for always having confidence in me. I would not have started this exciting journey if I had not met you.

To Yue Yan. You lightened me up in my most depressed moment and gave me the power to run through the finish line. You are the source of my strength with whom I am no longer afraid.

Last but most importantly, I would like to thank my parents Junxia Dai and Yuchun Dai. They gave me unconditioned love and support, and such great influence at early age that I started dreaming of being a researcher since kindergarten. They have never asked me to get good test scores, high GPA ranks, or many publications. Instead, they told me to be curious, be brave, work hard, and chase dreams. Therefore, I dedicate this dissertation to my wonderful parents.

Contents

List of Figures

List of Tables

Part I

Introduction and Background

Chapter 1

Introduction

We live in an information age where people are creating quintillion bytes of data every day. Information retrieval (IR) systems are tools for finding the needed information quickly from massive data collections. IR systems have become an essential part of people's daily lives – search engines like Google are receiving billions of searches per day, and almost all websites have a search bar for people to find articles, products, other people, etc. Information retrieval systems also play a critical role in other AI systems, finding useful data for downstream tasks such as data analysis, recommendation, question & answering, and dialogue generation.

A core task in information retrieval is to estimate the relevance between a query and a document. Modern information retrieval systems have relied on bag-of-words retrieval models, which count overlapping words between the query and the document. This simplified view of natural language allows the retrieval system to efficiently scan over millions or billions of documents, making large scale retrieval practical. However, counting overlapping words is only a shallow way of modeling search relevance. Intuitively, a better information retrieval system should be able to understand the meanings of the text and the semantic relation between queries and documents.

Improving language understanding in IR has indeed gained huge attention. However, despite considerable efforts, using more sophisticated natural language processing techniques in retrieval have been mostly unsuccessful. Recently, neural networks provide a powerful new paradigm for modeling natural language. *This dissertation aims to bring deeper language understanding into information retrieval using neural networks.* It focuses on two fundamental problems in information retrieval: text representation and relevance modeling. It discusses why these problems are difficult, develops novel neural network approaches to tackle the challenges, and demonstrates that the proposed methods successfully break the bottlenecks in previous state-of-the-art retrieval systems.

1.1 Bag-of-Words Retrieval

Bag-of-words retrieval models are the backbone of modern information retrieval. In essence, bag-of-words retrieval models make an intuitive assumption – a document is likely to be relevant to a query if it mentions the query words many times.

Bag-of-words retrieval models first break a piece of text into a set of independent words. For example, the query "machine learning and deep learning" can be represented as "{machine:1, learning:2, and:1, deep:1 }". The weight associated with each word aims to characterize the word's importance. Traditionally, the weight is usually calculated from term frequency statistics, such as term frequency in the current document (*tf*) and inverse document frequency in the collection (*idf*). Given the bag-of-words represen-

tations, the retrieval model then estimates the relevance between a query and a document by checking which words appear in both text, and leverages the matching words' weights to calculate a relevance score. That is to say, bag-of-words retrieval models represent queries and documents using *word-level frequency statistics*, and estimate relevance using *exact lexical match* between queries and documents. There are various bag-of-words retrieval models available differing in how they weight, normalize and combine the exact lexical match signals, to name a few, Boolean Retrieval, BM25 (Robertson and Walker, 1994), Statistical Query Language Models (Lafferty and Zhai, 2001), and Indri (Strohman et al., 2005).

A key advantage of bag-of-words retrieval models lies in efficiency. Statistics like *tf* and *idf* can be pre-computed offline, so that the retrieval system can look up the required word statistics without reading the actual document contents. More importantly, with a data structure called the inverted index, the retrieval system can skip documents that do not have overlapping words with the current query, significantly reducing the number of documents to be evaluated. Because of the simplicity and efficiency, bag-of-words retrieval algorithms such as BM25 has been the state-of-the-art retrieval models for decades.

The *learning-to-rank* systems (LeToR) (Qin et al., 2007) brought bag-of-words retrieval to a next level. With the increasing amount of available search data, it became possible to use machine-learned models to improve retrieval models, which were previously mainly based on heuristics. Learning-to-rank systems use the outputs of bag-of-words retrieval models as features, and applies machine learning to combine these features automatically. Other features, such as document qualities and query intent types, can also be incorporated into the machine learning model as additional evidence. First proposed in the year 2007, learning-to-rank systems have been state-of-the-art for over a decade. However, generating features and running machine learning models is often computationally expensive, limiting the learning-to-rank approach to only ranking a small set of documents.

Today, a full-fledged IR system usually consists of multiple ranking stages. It starts with a bag-of-words retrieval from the inverted index, which efficiently finds from a large collection (e.g., millions to billions of documents) a small set of candidates (e.g., dozens to thousands). Then, the system optionally applies one or more reranking stages to fine-tune and filter the initial results. As the candidate set only contains a small number of documents, the system can afford slower-but-more-accurate rankers, such as learning-to-rank models.

Bag-of-words approaches play a central role in the IR pipeline – it makes the initial large-scale retrieval possible, and provides key features to the learning-to-rank models in the reranking stages. However, bag-of-words is only a shallow way of understanding human languages, disregarding the meanings behind words and the context around words. It sets several limitations to current retrieval systems. Zhao (2012) suggested that two long-standing problems might be the bottlenecks of current retrieval systems.

- **Bottleneck 1: exact lexical match in query-document interaction**. Bag-of-words retrieval solely relies on exact lexical match. The machine only knows that a 'dog' document is relevant to the query 'dog', but does not know that a 'puppy' document is also relevant to the query 'dog' . Zhao (2012) show that on average, the vocabulary gap can be found in 40% to 50% of the relevant query-documents pairs. Thus, a better retrieval system should be able to draw connections between different words by considering their meanings in specific queries/documents
- **Bottleneck 2: frequency weighting in query/document representation**. State-of-the-art bag-of-words text representations uses term frequency statistics to estimate word importance. However, frequency does not necessarily reflect how important a word is to the meaning of the text. Zhao and Callan (2010) found that replacing the frequency-based weights with oracle weights can boost the retrieval accuracy for long queries by 50% to 260%. Thus, a better retrieval system should be able to understand a word's role in specific text, focus on words that are essential for the retrieval, and ignore words that are off-topic.

1.2 Neural Networks for IR: Advantages and Challenges

It has been almost thirty years since BM25 was proposed, and over a decade since learning-to-rank first came out. Over the past several decades, researchers have carried out extensive studies to bring more sophisticated natural language processing (NLP) techniques into information retrieval, but most of the attempts failed to come into real use.

A common theme of applying NLP to IR is to use linguistically motivated objects (part-of-speech tags, grammar-based parsers, word correlations, etc.) as derived from documents and queries, and create features or representations for retrieval. For example, dependency parsing (Metzler and Haas, 1989) and part-of-speech tagging (Allan and Raghavan, 2002) were studied for query/document understanding. Although inspiring, the parsers were fragile and limited to well-formed text, and the extracted linguistic signals often require additional, complicated processing to be useful for retrieval. Word co-occurrence is another NLP signal widely-studied in IR. For instance, there are extensive research on using statistical machine translation model (Berger and Lafferty, 1999) and topic models (Deerwester et al., 1990) in retrieval models. They allow the retrieval model to soft match two different words based on how often they co-occur. However, word co-occurrence patterns learned from the documents tend to behave differently from how people use words in retrieval tasks. In many cases, they only provide small gains over simple bag-of-words baselines. Moreover, the majority of existing NLP approaches treat text as sparse representations, making the models difficult to train and slow to use. For example, a classic machine translation model needs to learn a parameter for every possible word pairs. This problem is known as *the curse of dimensionality*.

Over the past decade, deep neural networks have shown astonishing success in many machine learning applications, offering a powerful new tool to model human languages. A deep neural network is a large set of simple mathematical units, called neurons, organize in layers, that can be trained together to complete complicated tasks. The layered structure enables the model to take the raw data as input and gradually learn high-level features. The ability to automatically discover complex features is one of the neural network's key advantages over traditional feature-based NLP techniques. Another strength of neural networks is the use of distributed representations. In contrast to using a single ID to represent a word, distributed representations convert words into dense vectors of a few hundred dimensions, addressing the "curse of dimensionality" problem faced by traditional machine learning models that use sparse representations.

However, in the field of information retrieval, the improvements made by neural networks appear relatively modest when compared to traditional techniques. Three vital challenges need to be addressed to make neural networks practical for information retrieval.

- **Challenge 1: Effectiveness**. At the time this thesis research started, most neural models failed to surpass feature-based learning-to-rank baselines. It remains to be answered how to design and train neural networks for information retrieval correctly.
- **Challenge 2: Efficiency**. Unlike bag-of-words retrieval models that can skip documents by word overlapping, neural networks use distributed text representations, which make a query comparable to all documents even if they do not mention queries terms. It is impractical to use neural networks during the early retrieval stages where millions or billions of documents need to be evaluated.
- **Challenge 3: Generalization ability**. Neural networks need to observe real queries and people's search behaviors to learn correct relevance patterns. Collecting such data is expensive and time-consuming. It is a question how neural IR solutions generalize to new queries, new documents, and new domains.

1.3 Thesis Research

This dissertation aims to leverage the advantages of neural networks to overcome the bottlenecks of bag-of-words retrieval, bringing deeper language understanding into today's information retrieval systems. We present a suite of neural network solutions to two long-standing problems in information retrieval. For query-document interaction, we study how to match words by their semantic relations. For query and document representation, we investigate how to weight words by their meanings and specific context. Our solutions combine advances in deep learning and NLP with insights from large-scale information retrieval, seeking to address effectiveness, efficiency, and generalization ability challenges in neural IR.

1.3.1 Query-Document Interaction: Neural Ranking Models for Soft Matching

One long-standing challenge in bag-of-words retrieval is the vocabulary mismatch problem (Furnas et al., 1987) – the retrieval model fails to match two words that are semantically related but lexically different. A word is a symbolic representation of certain underlying meanings. When seeing two different words, e.g., "puppy" and "dog", humans can quickly draw connections between their meanings. Bag-of-words retrieval models, on the other hand, lose those connections as it solely relies on *exact lexical match*. The first part of this dissertation seeks to bridge the vocabulary mismatch by developing a set of neural networks that can effectively *soft match* words in queries and documents.

As discussed earlier, one advantage of neural networks is that they represent a word as a distributed vector (embedding) rather than a discrete ID. With distributed embeddings, the machine can tell how similar two words are by simply taking a cosine similarity or dot product between the corresponding word embeddings. Although promising, previous attempts of using word embeddings in IR only achieved limited success. Word embeddings allow every word to match every other word, generating many weaker and noisier match signals that must be used carefully. How to use soft match signals effectively and reliably is an open problem. This thesis research developed `K-NRM`, a neural ranking model that uses a novel kernel-pooling technique to address these challenges. `K-NRM` uses multiple Gaussian kernels to separates the soft match similarities into different groups based on their contribution to relevance. For example, one kernel may capture near-synonyms such as ('dog', "puppy"), while another kernel may capture same-category concepts such as ('cat', 'dog'). A learning-to-rank layer combines the different types of soft match signals into a relevance score, where each kernel contributes differently to the final score. `K-NRM` was used at the reranking stage to fine-tune the document ranking from an initial retrieval. It is the first neural ranker to outperform strong learning-to-rank baselines that held the state-of-the-art at the time, revealing the power and potential of soft match signals for IR.

Following `K-NRM`, this dissertation proposes `Conv-KNRM`, a convolutional kernel-based neural ranking model for soft matching n-grams. N-grams have been widely used in traditional bag-of-words retrieval to preserve phrases and short-term dependencies, but they are usually treated as discrete terms and matched exactly. For example, a bi-gram "white house" from a document can only match to "white house" in queries. `Conv-KNRM` aims to let search engine also soft match n-grams, for example, "white house" and "US President". `Conv-KNRM` approaches to this goal by employing Convolutional Neural Networks (CNN) to compose adjacent words' embeddings to n-gram embeddings, and then using the kernel-pooling technique to learn different groups of n-gram soft match. The CNN is the key to modeling n-grams. Treating n-grams atomically will explode the parameter space and suffer from data sparsity. `Conv-KNRM` uses CNNs to generate n-grams on-the-fly and re-uses knowledge about individual words to represent n-grams. Besides, the convolutional layer projects all n-grams into a unified embedding space, so that short n-grams can match long n-grams. For instance, "white house" in the document can provide partial evidence for the

query "George Walker Bush". Soft match of n-grams is a new type of signal that has not been explored in previous research. Conv-KNRM makes it possible, efficient, and effective.

This dissertation research also investigated the generalization ability of K-NRM and Conv-KNRM architectures to other data modalities. Neural networks, with the use of distributed representation and the ability to learn from scratch, make it possible to use similar architectures for a variety of search tasks that involve different modality of data, such as images and music. We believe that different search tasks share a set of underlying problems, for example, how to model soft match. Under distributed representations, queries, and documents, no matter they are text, codes, tables, or even images, can be converted into a unified representation, enabling the model to deal with a universal representation without worrying about the original data type. With end-to-end learning, features can be learned automatically, significantly reducing the efforts of domain-specific feature engineering. This dissertation adopts the K-NRM and Conv-KNRM models to searching engineering diagram – an image search task – and shows that these models address some fundamental problems across different search tasks.

The effectiveness of K-NRM and Conv-KNRM mainly comes from their IR-customized soft match, which need to be learned by observing how people search, e.g., from search logs or expert annotations. Those data are expensive to collect and may raise concerns on privacy. Meanwhile, some soft match patterns, such as the match between "atypical squamous cells" and "cervical cancer", can be learned from the documents without observing search logs. It leads us to develop the DocBERT reranker, which seeks to enhance neural ranking models with general language understanding knowledge from a large pre-trained language model BERT (Lee et al., 2019a). The DocBERT reranker uses a passage-based ranking framework to address the input length limitation in BERT, making it possible to rank long documents. We show that this architecture is effective, and the general-purpose pre-training can greatly compensate for lack of training data in low-resource conditions. The domain adaption technique and the general language modeling can stack up and further improve search effectiveness, showing that an effective neural ranking model should be equipped with two types of knowledge about human language: how people use languages when searching, and how people use languages in general. Further analysis investigates the source of effectiveness of DocBERT, and finds that in addition to model query-document soft match, DocBERT also uses self-attention to capture sentence structures and build context within a query and a document. It helps DocBERT on several problems that were considered difficult in traditional IR, such as prepositional phrases, long queries, stopwords and punctuation.

In summary, the first part of this thesis research focused on modeling the *interactions* between queries and documents. It developed novel neural ranking models that can soft match words, bridging the vocabulary gap between queries and documents. Meanwhile, DocBERT reranker reveals new possibilities in understanding the text content of a query or a document. It led us to the next part of this thesis research, which seeks to improve the *representations* of queries or documents by leveraging the language understanding ability from deep neural networks.

1.3.2 Query/Document Representation: Context-Aware Term Importance Weighting

While deep learning models have drawn increasing popularity in the reranking stage, the initial retrieval stage in most retrieval systems still relies on older bag-of-words retrieval models like BM25 (Robertson and Walker, 1994). A key reason is efficiency. Bag-of-words text representations can be stored into the inverted index that allows the retrieval model to skip documents that have no overlapping words with the query, and are therefore efficient enough to serve in the *initial retrieval* stage – retrieve and rank over the entire corpus.

The common practice in bag-of-words retrieval models is to weight words using *term frequency* statistics, such as term frequency in the document (*tf*) and inverse document frequency in the collection (*idf*).

Term frequencies achieve a reasonable level of accuracy, but they do not fully reveal whether a term is essential to the meaning of a query or a document. An example would be the following paragraph:

> *"Unlike cats, dogs are usually great exercise pals. Many breeds enjoy running and hiking, and will happily trek along on any trip. "*

As humans, we understand that the paragraph is more about "dog" than "cat". However, both "cat" and "dog" appear once in the paragraph, so the retrieval model is likely to treat them as equally important, and therefore mistakenly return this documents for a "cat" query.

The DocBERT reranker developed in the previous part of this dissertation shows that deep language models like BERT are capable of using the current linguistic context to understand a word better. This finding motivates the second part of this dissertation – *using better language understanding techniques to generate context-aware bag-of-words text representations that improve initial retrieval.* Our methods estimate term importance based on its contextualized embedding from a deep language model, providing a new alternative to the long-used *tf* weighting. More importantly, the resulting text representations are still in the form of simple bag-of-words that can be inverted indexed, ensuring efficiency for retrieving over large corpora.

We first present DeepCT, a Deep Contextual Term Weighting Framework for sentences and short documents. Traditional term frequency based retrieval often faces challenges in short text as the frequency distribution is flat. DeepCT addresses this challenge by estimating term importance from its meaning and role in the text. It first uses a deep language model – such as BERT – to generate contextualized embeddings for words in a text. The embedding not only encodes a word's type and meaning, but also its syntactic and semantic features in the current text. Next, DeepCT learns to map these embeddings into context-aware term weights. To mitigate the difficulty of manually annotating words importance, we propose a distance supervision method that mines the training labels from relevance-based query-document pairs. DeepCT is applied at the offline index time, generating bag-of-words document representations that can be stored in a typical inverted index. To use the new index is as simple as replacing *tf* in a standard retrieval model with the new context-aware term weights. On two passage retrieval tasks, DeepCT improves the accuracy of two popular bag-of-words retrieval algorithms significantly.

Next, we present HDCT, a Context-aware Hierarchical Document Term Weighting Framework that extends DeepCT from short passages to long documents. With our current implementation, DeepCT cannot be straightforwardly applied to long documents as long text input will exceed the memory limitation. HDCT uses a hierarchical approach to build document representations from passage ones. It uses the DeepCT framework to estimate a term's local importance in each passage, and combines the passage-specific term weights into a document bag-of-words representation. Trained on document-level term importance labels, HDCT not only learns which terms are essential in the passage, but also determines which passages are critical to the document. To improve the generalization ability of HDCT, we developed a content-based weak-supervision strategy that does not need any query logs for training. It automatically generates training labels by exploiting the internal structure of documents, without using any external labels or data. We show that a model trained on the content-based weak-supervision signals can be the same or even more accurate than supervised models under low resource conditions.

We view DeepCT and HDCT as an encouraging step from "frequencies" to "meanings'. DeepCT can find the most central words in a text even if they are mentioned only once; it also suppresses non-central words even if they appear frequently. In HDCT, document-level key terms are promoted, diverse topics from different passages are revealed, while unimportant words and noisy passages are depressed. Such deep, fine-grain text understanding leads to significant improvements to the standard bag-of-words retrieval models on several benchmark document retrieval datasets, establishing new state-of-the-art for initial retrieval problems.

1.4 Significance of the Research

This dissertation develops a suite of neural network solutions to improve language understanding in information retrieval.

- For the reranking stages, we developed a set of neural ranking models (`K-NRM`, `Conv-KNRM`, and the `DocBERT` reranker) that accurately soft match queries and documents.
- For the initial retrieval stage, we developed two neural bag-of-words models (`DeepCT` and `HDCT`) that produce context-aware term weights for inverted indexing and efficient retrieval.

The effectiveness of the proposed methods has been demonstrated generally across various scenarios using highly competitive academic benchmarks. Besides a set of state-of-the-art neural IR models, this dissertation research also provides the research field with several impactful research findings.

This dissertation research gives a new understanding of how language should be modeled in IR. It shows that language patterns extracted from corpus analysis can be quite different from how people use languages when searching the corpus. From earlier graph-based methods in the 1980s to recent word embeddings, researchers have tried many ways to bridge the vocabulary gap between queries and documents. Still, few of them managed to surpass a learning-to-rank baseline that only used exact lexical match features. Previous research assumed that words with similar linguistic uses in a corpus are also relevant in search. Our research proves that such an assumption is not correct – `K-NRM` decouples over 90% of word pairs that are considered similar in word2vec. The newly discovered soft match patterns encode search-specific patterns that have not been well understood yet, and are worth studying in future research.

This dissertation presents several methods to train neural IR models under low resource conditions. We argue that an effective search engine should acquire both knowledge about general human language and knowledge about the search task. Our research show that pre-trained language models can effectively provide general language knowledge. For search-specific knowledge, we developed several weak-supervision approaches that reduce the number of search logs needed, including borrowing search logs from related domains, generating pseudo-labels using a weaker search engine, and mining query-like signals from the document content. Together, these approaches will help IR researcher and developers to handle cold-start scenarios and low resource domains. In addition, we demonstrate that different search domains, e.g., text and image, share several common issues, and it is possible to design a universal neural ranking model to handle these issues regardless of the differences in data modalities. The landscape this thesis research sets up provides a wide range of opportunities for future research in learning generalizable retrieval models.

This dissertation research provides a new deep-learning based framework for indexing and retrieval. People have been using term frequencies to index documents for decades. `DeepCT` and `HDCT` shows that term frequencies are no longer sufficient. By leveraging deep language models to capture a word's context, our models generate a novel semantic-based term weights, which improved the initial retrieval accuracy by up to 40%. The results indicate that the initial retrieval stage have large potential for improvements with recent deep learning techniques. More broadly, this dissertation shows that neural models have two distinct contributions to more accurate retrieval: *text understanding, and matching*. While the matching needs to be done online, the text understanding part can be done offline, opening up many opportunities for the next-generation indexing and retrieval system. With the lessons learned from `DeepCT` and `HDCT`, now we are free to think of the indexing process as a balanced trade off between efficient text representations and deep language understanding.

This dissertation formulates a new neural network based retrieval paradigm that overcomes the limitation of bag-of-words and frequency based retrieval. It points out several promising paths in neural

relevance ranking, neural retrieval models, and deep document understanding for IR. We believe it will inspire more Ph.D. dissertation research in the near future.

1.5 Organization

The rest of this dissertation organizes as follows. Chapter 2 discusses the background. Part II presents our work on neural ranking models for soft matching quires and documents. It includes research results on K-NRM (Chapter 3), Conv-KNRM (Chapter 4), and the DocBERT reranker (Chapter 6). Part III presents our work on neural bag-of-words representations with context-aware term weights. It includes research results on DeepCT (Chapter 7) and HDCT (Chapter 8). The last part (Chapter 9) summarizes and discusses the impacts of this thesis research.

Chapter 2

Prior Research

This chapter first provides background knowledge about today's information retrieval (IR) systems. Next, it discusses prior approaches to improving language understanding in IR and their limitations. Finally, it introduces deep neural networks, discusses their advantages over prior approaches, and reviews prior work that applies neural networks to IR.

2.1 Today's Information Retrieval Systems

A modern information retrieval systems is typically composed of three high level stages, as shown in Figure 2.1.

- Indexing: an offline stage that processes and stores documents to facilitate fast and accurate retrieval.
- Initial Retrieval: when a user gives a search query, this stage retrieves a set of candidate documents from the index. The retrieval algorithms used in this stage must be efficient as it needs to compute the retrieval score for all the documents in a large-scale collection, The retrieved documents can be directly presented to the user, or sent to the next stage – the reranking stage.
- Reranking: it involves one or more steps that gradually filters and refines the candidate list using slower but more accurate methods.

The indexing step collects, parses, and stores documents to facilitate retrieval. There are two common types of indexes: *inverted index* and *forward index*. The inverted index maps back from a *term* – a tokenized and normalized word – to a list of documents where the term occurs. Essentially, this is an efficient way of storing bag-of-words document representations, which, as will be discussed later, helps initial retrieval models run fast. The forward index maps a document ID to the document's content, including the raw text as well as processed ones like bag-of-words representations. The forward index is usually used in the reranking stage where the ranking models need access to the document content to extract features.

The common practice for the initial retrieval stage is to use a bag-of-words retrieval model such as Boolean retrieval, BM25 (Robertson and Walker, 1994), or statistical language model (Lafferty and Zhai, 2001). They assign a relevance score for a document with respect to a query relying mostly on the level of matching between the query terms and the document terms. This type of retrieval models can process queries very fast as the inverted index allows them to quickly find documents that mention the terms and skip all other documents that do not have overlapping words with the query.

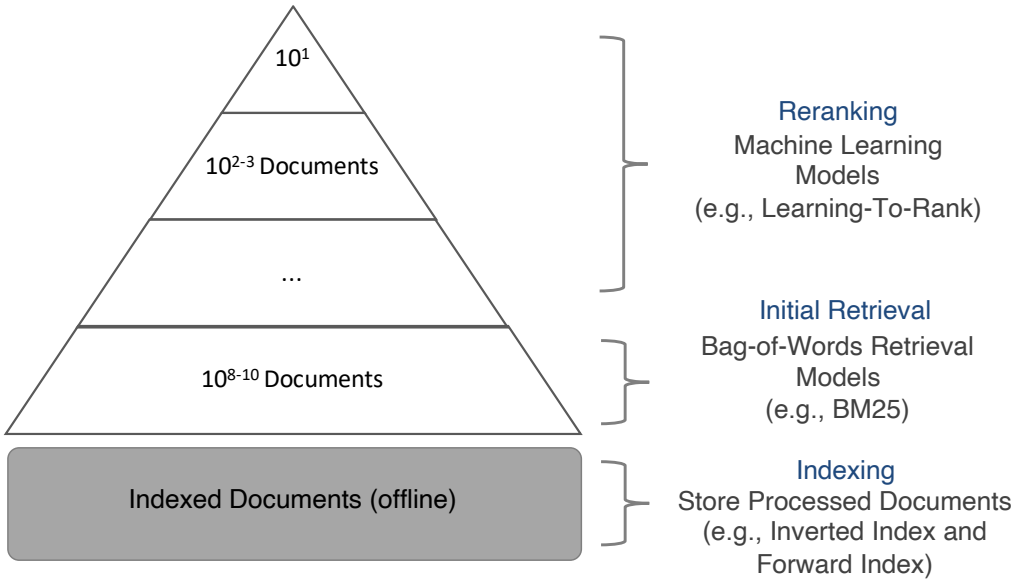


Figure 2.1: A typical multi-stage information retrieval pipeline.

.

The most simple form of bag-of-words retrieval model is the unranked Boolean retrieval, which checks whether or not the documents contain the query terms and satisfy Boolean constraints. Later retrieval models, such as ranked Boolean, BM25, and statisitcal language models, improve unranked Boolean by considering how often the documents mention the query terms. These retrieval models share a similar *tf.idf*-style formula. *tf*, short for term frequency, is the frequency of a term in a document; it estimates the local importance of a term in a document. *idf*, the inverse document frequency, is the inverse of the occurrence frequency of a term in the whole document collection; it estimates the global importance of a term in a collection, assuming that rare terms are generally more discriminate. Fundamentally, these retrieval models are based on term frequency statistics without considering the semantics of words.

The search engine can take the top ranked documents from the initial ranking and send them to one or more reranking stages to fine-tune the results. The rerankers use slower but more accurate algorithms to filter and rerank them. The most widely used re-rankers are *learning-to-rank* systems (Qin et al., 2007).

Learning-to-rank refers to machine-learned ranking models that automatically learn how to combine evidence for ranking. It uses handcrafted features extracted from the query and the document, and learns models that combine these features into a relevance score. Commonly used features can be categorized into query-document features, document-only features, and query-only features. Query-document features characterize the interactions between the query and the document. Typically, these are from basic bag-of-words retrieval models such as Boolean retrieval models, BM25, and query language model. Document-only and query-only features capture the document's or the query's own characteristics, such as quality and popularity estimates. Modern learning-to-rank systems usually use dozens to hundreds of pre-defined features. The diverse evidence and machine-learned models make learning-to-rank much more expressive and aligned with the retrieval goals. Nevertheless, features are based on human intuition that can be incorrect or incomplete. The majority of effective query-document features are based on bag-of-words and *tf.idf* weights, i.e., they are variations on a single idea. The reranking model is still missing semantic evidence.

Inverted indexing, initial retrieval and re-ranking have shaped today's retrieval systems. Bag-of-words approaches plays a critical role at every stage of the pipeline – the index stores documents by individual terms, the retrieval model counts the matches between query terms and document terms, and the reranking model relies on the output of bag-of-words retrieval models as features to describe query-document interactions. Such pipelined systems have been state-of-the-art for the past decade and are widely used in the industry, e.g., Yahoo! (Yin et al., 2016) and Yandex (Trofimov et al., 2012).

Although being very successful, bag-of-words retrieval sets several limitations to the retrieval systems. They have difficulty retrieving relevant documents that have very low frequency of, or even absent of, the query terms. This problem is known as the *vocabulary mismatch problem* in which a relevant document uses terms that are related to, but are not the same as the query terms. Even when the query terms do appear in the documents, bag-of-words retrieval models still face the challenge of understanding how important a term is to the relevance. Current *tf.idf*-style methods weight terms solely based on how frequency a term occurs, which may not align well with their semantic importance in the text.

Going beyond bag-of-words, a next-generation retrieval system should be capable of understanding the meanings of documents and queries, and rank documents accurately based on their semantic relations to the query. Such a system may use completely different text representations and ranking models from the current ones. This thesis research investigates one potential direction – neural network based retrieval systems.

2.2 Prior Work to Improve Language Understanding in IR

Much research has focused on improving bag-of-words retrieval models. Some of them are very successful. Sequential dependency model (SDM) (Metzler and Croft, 2005) represents a text not only by terms but also by pairs of terms that co-occur in within a distance, capturing short-range dependencies between terms. Several retrieval models use controlled vocabularies to map the raw text into a well-defined ontology, so that documents can be indexed and searched by pre-defined entities and categories (Rajashekar and Croft, 1995; Lu et al., 2009; Xiong et al., 2016). (Pseudo-)relevance feedback techniques expand the original query with terms from (pseudo-)relevant documents, to better define the search target and to compensate for the vocabulary gap between queries and documents (Lavrenko and Croft, 2001). Multi-fielded bag-of-words, such as BM25F (Zaragoza et al., 2004), store different document fields separately, allowing the retrieval model to give higher weights to more important fields.

The methods mentioned above have been widely adopted in academia research and industrial applications, but they are still within the bag-of-words framework – they rely on frequencies to weight index terms and have to match terms exactly. Over the past decades, researchers have carried out extensive studies to bring more sophisticated natural language processing (NLP) techniques into information retrieval.

One direction focused on the use of language syntactic structures, such as grammar trees and part-of-speech tags. Metzler and Haas (1989) parsed the queries and documents into dependency trees, and designed rules to match the dependency trees. The overall improvements in retrieval effectiveness were relatively small and inconsistent, because the rule-based system is not robust to low-quality text input. Strzalkowski (1995) used grammar trees to identify phrases in documents and use them as new index terms. This method was later surpassed by the sequential dependency model (SDM (Metzler and Croft, 2005)) that is more flexible. Allan and Raghavan (2002) used part-of-speech patterns to address query ambiguity. They extract part-of-speech patterns, map them to manually generated clarification question templates, and present those questions to the user for clarified queries. The problem with many of these techniques is that they are high in precision, but low in recall. They make the query very sensitive to the

syntactic structure of a relevant sentence and can only match within sentences. The empirical evidence was sparse and limited to small-scale tests.

Another line of research focused on the semantic relationships between words – usually modeled by counting how two words co-occur in the language. One way to do so is using graph-based methods. They model represent a text as a graph, where nodes represent words and edges represent co-occurrence of words within a distance. The retrieval model can use the graph to rank words and identify phrases (Mihalcea and Tarau, 2004) using algorithms like PageRank (Page et al., 1999). Latent topic models have also received much attention. Latent Semantic Indexing (LSI) (Deerwester et al., 1990) and Latent Dirichlet Allocation (LDA) (Blei et al., 2001) seeks to discover latent topics in the text, and match the query and the document by the latent topics. These approaches were mostly unsupervised. They capture words that share similar context in documents, but ignored the fact that language usages in searching documents can be different from those in writing documents. A common error found in these approaches is that they consider "cat-dog" to be a match, but people searching for "dog" are unlikely to be satisfied with documents about "cat".

Statistical translation models are a supervised approach to modeling word semantic relationships. They model query-document relevance using a pre-computed translation matrix that describes the similarities between word pairs (Berger and Lafferty, 1999). At query time, the ranking function considers the similarities of all query and document word pairs, allowing query words to be soft matched to document words. The translation matrix can be calculated via mutual information in a corpus (Karimzadehgan and Zhai, 2010; Jin et al., 2002) or using user clicks (Gao et al., 2010). A critical limitation in statistical translation models is that the translation matrix faces the *the curse of dimensionality*. For a corpus with $|V|$ words, the model needs to learn $|V^2|$ translation scores to cover every pair of words. The parameters grow even larger when phrases and n-grams are taken into consideration. It makes statistical translation models difficult to train at a large scale, and often fail to outperform a standard BM25.

To summarize, this section briefly discusses prior research that uses linguistic rules and statistical learning methods to improve language understanding in IR. While those methods were capable of capturing certain syntactic and semantics, their capability was limited for several reasons. The handcraft rules have a limited coverage, causing retrieval models to only focus on a small set of matchings that fit the rules. The statistical methods often draw statistics from the documents, disregarding the specific patterns people use when searching. Even when the statistics are estimated from search-specific data (e.g., some statistical translation models), the model often faces the curse of dimensionality raised from the high-dimension, discrete representation of the text. Human languages have diverse uses and are by nature a symbolic system that is high-dimensional, making them generally difficult to model using rules or traditional machine learning techniques. In addition, to model languages in search has additional unique challenges, as searching a large corpus can be time-consuming, and people have not yet understood well how a query interacts with the documents.

2.3 Neural Networks and Their Applications to IR

Neural networks are a special type of machine learning model. A neural network is a set of simple math units, called neurons, organized into layers, to complete complicated tasks. The universal approximation theorem (Csáji, 2001) states that a simple neural network with a single hidden layer containing a finite number of neurons can approximate continuous functions on compact subsets of R^n, meaning that neural networks can represent a wide variety of interesting functions when given appropriate parameters. The universal approximation property makes neural networks different from most other existing machine learning models, and therefore has attracted several waves of research on neural networks.

The idea of neural networks began almost 80 years ago as an electrical model of how neurons in the brain function. Neural networks have evolved significantly since then. A burst of neural network research happened during the 1980s and 1990s – back-propagation was developed to train neural networks; particular neural networks architecture, such as convolutional neural networks and recurrent neural networks, were designed to model spatial and sequential patterns. This burst of research also attracted researchers from the information retrieval field, and led to the first wave of neural IR research. Back then, most studies on neural networks were limited to toy settings that use small datasets and small models.

In the recent decade, big data and fast computer processors brought neural networks into the era of *deep learning*. Larger neural networks were developed, and more training data were used. Neural networks began to outperform traditional machine learning methods in several applications, such as visual object recognition and speech recognition. Deep models and big data unleashed three advantages of neural networks over traditional machine learning algorithms.

- Neural networks have the expressive power to represent complex functions. These functions are often too complicated and too many to be described using handcraft features.
- Neural networks have the capacity to learn from large data, fitting the hidden correlations in complicated tasks. People look for features and write programs to extract them; oftentimes, the features do not fully align with the end task. Neural networks are trained to fit the final task, bridging the gap between features and objectives.
- Neural networks uses low-dimensional, distributed representations. Distributed representations convert discrete data, such as words, into low-dimensional dense vectors, avoiding the curse of dimensionality problem faced by traditional machine learning models that use sparse representations.

These advantages of neural networks offer a new paradigm for modeling language in information retrieval, leading to the second wave of neural IR research. The second wave of neural IR research began with extending classic retrieval models to incorporate word similarities learned from neural networks, with the goal to bridge the vocabulary gap between queries and documents. It then shifted towards developing new retrieval models that directly use neural networks to estimate relevance scores, motivated by that neural networks can automatically discover relevance patterns from query-document pairs. Recently, researchers started to explore using neural networks in indexing and initial retrieval, where efficiency constraints are crucial.

Neural IR is a rapidly developing area, Many new related work has emerged in the same time as this thesis research is conducted. This sections aims to provide a background knowledge about prior work and explain how they motivate this thesis research. Prior work that is related to specific chapter will be discussed separately within the chapter.

The rest of this section starts with reviewing the first wave of neural IR research from the 1980s-1990s (Section 2.3.1). Then it focuses on recent neural IR research, including work that incorporates word embeddings into classic IR models (Section 2.3.2), neural ranking models that are used to re-ranking a few documents (Section 2.3.3), and neural retrieval models that are applied to the indexing and initial retrieval stages (Section 2.3.4).

2.3.1 Early Neural IR

The 1980s - 1990s witnessed the first wave of research on applying neural networks to information retrieval.

In 1986, Belew (1986) published his Ph.D. dissertation *'Adaptive information retrieval: machine learning in associate networks'*. The technique proposed in this dissertation was more similar to today's graph-based approaches – using a neuron (node) to represent text, and using the connections between neu-

rons to represent their associations. An improved version of the system, called AIR, was later published as a conference paper (Belew, 1989). AIR has three types of neurons (document, author, and keywords) and two types of links (document-author, and document-keywords). The links are weighted according to an inverse frequency weighting scheme. After the user submits a query, a sub-graph is shown to the user with nodes that are related to query. Using a graph user interface, the user can mark useful nodes that are related to the query. The feedback is then used to update the link weights and generate a new query. The feedback allows the system to improve with use; but at that time, the updating algorithm is based on specially designed rules rather than a general-purpose optimization algorithm such as back-propagation.

In 1989, Kwok (1989) first demonstrated that a neural network could provide retrieval effectiveness similar to that of the probabilistic indexing and retrieval model based on single terms. The network (Kwok, 1989) used a similar graph-based architecture as AIR (Belew, 1989). It consists of a query layer, a term layer, and a document layer. A query is represented as a neuron; it is connected to its query terms in the term layer; the term neurons are then connected to documents that they appear in. The connection weights between neurons are set to mimic a probabilistic retrieval model: query term frequency for query-term links and a *tf.idf* weight. During retrieval, the activity of the query neuron is turned on. The activation spreads through each query term neuron, reaches the documents, and generates the relevance score. Kwok's 3-layer architecture was flexible and well suited for the IR task. This architecture became a general framework that was widely used by following research.

Kwok's approach was essentially a re-implementation of the *tf.idf* retrieval model using a neural network. It only supports exact lexical match, but does not bring in new features (e.g., semantic match). The MERCURE system (Boughanem and Soulé-Dupuis, 1994) was one of the first systems that combined the term associations and adaptive learning for general domain IR. MERCURE adopted a 3-layer network architecture similar to Kwok's approach. Differently, terms are also interconnected with each other in MERCURE. The term nodes are connected by co-occurrence links based on the assumption that if two terms co-occur in many documents then these terms are likely to be semantically related. During weight propagation, the activated nodes spread their activity to other term nodes and imply an automatic query expansion. The authors also proposed a short-term learning process that adjusts the weights of co-occurrence links according to a user's relevance judgments on retrieved documents. This work was novel in using co-occurence links to model semantic relations between terms, which gives the model the ability to go beyond exact lexical match.

Most of the early neural IR research mentioned above used handcrafted rules to update the model when receiving new user feedback. (Wong et al., 1993) was one of the first neural IR model that used gradient descent do optimzie the neural netwrks. Experiments were conducted on a small-scale dataset of 85 documents, 35 queries, and 1217 index terms. The neural network learned from relevance feedback improved a simple vector space model by 60%. One main limitation of the approach from (Wong et al., 1993) is the curse of dimensionality – the term connection weight matrix needs to learn n^2 weights for a vocabulary of n terms.

Gallant proposed an interesting work that used low-dimensional, distributed word representation as opposed to the discrete representations (Caid et al., 1995). This method, called MatchPlus, represents a word using a real-valued vector. Words that occur in similar contexts have vector representations near each other. MatchPlus is similar to Latent Semantic Indexing (LSI) (Deerwester et al., 1990) in that words are represented by dense vectors. Differently, MatchPlus uses local context, i.e., nearby words, to generate term vectors rather than document-wide context as in LSI. The use of local context relates MatchPlus to more recent context-based word embedding techniques like word2vec (Mikolov et al., 2013). MatchPlus then forms query and document vectors as the weighted-sum of the word vectors for words contained in the query or document; retrieval is through finding the document vector that maximizes the inner product

to the query vector. The authors compared MatchPlus to an LSI-based retrieval system. Mixed results were observed: MatchPlus had higher precision at shallow rankings but lower average precision at deeper rankings. Both MatchPlus and LSI based systems were weaker than bag-of-words *tf.idf* methods.

In summary, the popularity of neural networks in the late 1980's caught the attention of the IR research community. Researchers identified two main advantages of neural networks for information retrieval tasks: *modeling term associations* and *learning from user feedback*. These advantages have the potential to break the limitations of exact word-match based retrieval models and heuristic ways of deciding model weights. These two advantages still hold for today's information retrieval systems. However, the earlier works only achieved limited success due to several drawbacks. First, most of the networks assign a node to every single index term. The large amount of nodes makes the network slow and hard to use in a real application. Second, the learning ability of neural networks were not fully exploited. Large-scale search logs were less available at that time, so most of the methods build the weights from the corpus using already existing formulas such as *tf.idf*. Moreover, the standard computing hardware at that time did not support learning with large data. With these obstacles, the first wave neural IR methods were not more effective than simpler approaches.

2.3.2 Bridging Vocabulary Mismatch with Word Embeddings

In the last decade, a significant evolution of deep learning has taken place. Thanks to the availability of a larger amount of data, faster computing hardware, and better training techniques, neural networks started to show learning capacity that had not been not fully exploited before. An important milestone of applying neural networks to natural language processing is the development of neural network learned distributed word representations, also called *word embeddings*.

Word embeddings came into massive attention when word2vec (Mikolov et al., 2013) was proposed. Word embeddings represent a word as a low-dimensional dense vector. The embeddings are learned from a large corpus to predict a word based on its context using a shallow neural network. This is based on the distributional hypothesis that words appearing within similar context possess similar meaning. This was not a new idea – MatchPlus (Caid et al., 1995) essentially used the same approach. Word2vec became popular because it uses new techniques (e.g., stochastic gradient descent, negative sampling, etc) that can efficiently construct high-quality word embeddings for larger corpus, making the method fast, scalable, and easy to use. Word2vec has inspired many follow-on works, such as the Global Vectors for Word Representation (GloVe) proposed by Pennington et al. (2014), which aims to explicitly approximate the global co-occurrence countings using word embedding, and FastText embedding (Mikolov et al., 2017), which take subwords (Schütze, 1993) into account to better handle unseen words.

The development of word embeddings attracted widespread interest from the IR research community. Much research has been carried out on using word embeddings for IR. Most of these studies focused on using word embeddings to find similar words in order to bridge the vocabulary gap between queries and documents (the vocabulary mismatch problem). This section gives an overview of these studies.

One way to use word embeddings for IR is to incorporate them into statistical translation models. In information retrieval, statistical translation models treat queries and documents as two different languages, and estimate their relevance as the translation probabilities. Conventional translation models suffered from sparsity because they need to learn a probability for every possible word pair. Word embeddings provide a smooth low-level approximation of word similarities. The Neural Language Translation Model (NLTM) (Zuccon et al., 2015) estimates the translation probability using the normalized cosine similarity of word embeddings. Empirical results indicate that the NLTM model has moderate improvements over the conventional mutual-information based translation models. A similar approach is the Generalized Language Model (GLM) (Ganguly et al., 2015), which integrates word embeddings into a classic query-

likelihood language model. In GLM, the probability of observing a term t in the query from a document is modeled by two distinct events, that of generating a different term t', either from the document itself or from the collection, and then transforming t' to the observed query term t. The transformation probability is derived from the similarities between the corresponding word embeddings. Superior performance over the query-likelihood model and Latent Dirichlet Allocation (LDA) was reported.

The similarity between word vectors reflects the similarity between two terms, making word embeddings also a good fit for query expansion. Roy et al. (2016) proposed a method that select expansion terms by their cosine similarity to the query terms calculated from corpus-trained word2vec. The expansion terms are pruned by only keeping terms that appear in the query-specific pseudo-relevance documents. Experiments show that the query-specific pruning improved the quality of expansion terms, indicating the necessity of topic-specific statistics.

Diaz et al. (2016) compared the effectiveness of corpus-trained global word embeddings and query-specific local word embeddings on query expansion. The local word embedding is trained on a sample of pseudo-relevant documents that are retrieved for the query on high-quality external corpora (Gigaword and Wikipedia). The local word embeddings are then used to derive an expansion language model for the query. The results show that query-specific word embeddings outperform the global word embeddings due to the ability to capture topic-specific statistics. The results raised questions about whether corpus-based word embeddings are aligned with the ad-hoc search task.

With many attempts to use word embeddings to address the vocabulary mismatch problem, only moderate and local improvements over traditional retrieval models have been observed. Training distributed word representation using the surrounding context was not an entirely new idea. There is a long history of distributed word representations. Examples include Latent Dirichlet Allocation (LDA) (Blei et al., 2001), Latent Semantic Indexing (LSI) (Deerwester et al., 1990), and MatchPlus (Caid et al., 1995). A study from Levy et al. (2015) argued that the mathematical objects and the sources of information available to word2vec are in fact very similar to those employed by the traditional methods, and reported only insignificant performance differences between the methods, with no global advantage to any single approach over the others.

The mixed results achieved by the above attempts raised questions on the suitability of context-based word embeddings for ad-hoc search. Ai et al. (2016) showed that context-based embeddings are insufficient for modeling word substitution where two words are related but may not co-occur in documents, such as "clothing" and "garment". Rekabsaz et al. (2017) demonstrated that word embeddings might bias the query to unrelated topics. For instance, antonyms ("cheap" and "expensive") or co-hyponyms ("mathematics" and "physics") share common window-context and are therefore considered similar in the word embedding space, but using such word pairs for substitution or expansion will shift the focus of query to unrelated topics.

Some studies tried to filter the noisy soft match signals in context-based word embeddings. As mentioned earlier, on the query expansion task, Roy et al. (2016) pruned the expansion terms found by word embedding similarities by limiting the candidate terms to appear in the pseudo-feedback documents. On document ranking task, Rekabsaz et al. (2017) filtered the related word pairs using the document-level word co-occurrence statistics, assuming that effective word pairs should not only have similar window-context but also have similar document-context. Later, Neural IR research started to explore IR-customized word embeddings. Diaz et al. (2016) train word embeddings using pseudo-relevance feedback (PRF) documents for each individual query. This approach is effective but faces efficiency challenges because it trains a new word2vec every time it receives a new query. Zamani et al. (2018b) proposed a more efficient way to train PRF-based embedding. The word embedding is trained on millions of (query, retrieved documents) pairs. The information encoded in the word embedding is no longer

window-context information, but the relationship between queries and their retrieved documents. Experiments show the superiority of relevance-based word embedding over context-based word embedding on query expansion and query classification.

2.3.3 Neural Models for Ranking

Incorporating word embeddings into traditional retrieval models only achieved insignificant improvements, suggesting the need for more IR-customized word embeddings and neural models. On the other hand, the success of learning-to-rank systems demonstrated the power of supervised learning from relevance feedback data. It inspired researchers to shift the focus of applying embeddings to existing retrieval models towards developing new neural ranking models.

Neural ranking models are a special type of learning-to-rank system (Qin et al., 2007) that use neural networks to generate and combine features. They differ from traditional learning-to-rank systems in the following ways. First, neural ranking models use distributed representations to represent the content, which could be more efficient than sparse representations. Second, neural ranking models use layers of simple non-linear functions to transform and summarize the low-level input into high-level features. It makes it possible to use raw text as input, lifting the developers' burden of handcrafting features. Based on how they represent and match queries and documents, the current neural ranking models can be categorized into two classes: *representation-based* and *interaction-based* (Guo et al., 2016a).

Representation-based neural ranking models follow the idea of Siamese network, which was first introduced in the early 1990s by Bromley et al. (1993) to solve signature verification problem. A Siamese neural network consists of twin networks that accept distinct inputs but are joined by an energy function at the top. This function computes some metric between the highest-level feature representation on each side. Representation-based neural ranking models independently learns a representation for the query and the document and then calculates the similarity between the two estimated representations via a simple similarity function. This line of research can be traced back to research in conventional latent semantic modeling. For example, LSI (Deerwester et al., 1990) decomposes a document-term matrix and learns a linear projection that casts bag-of-words query/document vectors into a latent space. The relevance score between a query and a document is assumed to be proportional to their cosine similarity of the corresponding dense vectors. LDA (Blei et al., 2001) represents text by their latent topic distributions. Wei and Croft (2006) proposed an LDA-based retrieval model that computes relevance scores using the similarity between the topic distributions of queries and documents. One of the earliest representation-based neural ranking models is the MatchPlus system (Caid et al., 1995). This work is discussed in Section 2.1. Here we focus on recent works in the second wave of neural IR research.

Salakhutdinov and Hinton (2009) proposed a neural IR model that employs deep auto-encoders for semantic modeling. They use a multi-layer auto-encoder to learn distributed representations of documents. The paper demonstrated that deep neural networks are able to discover the hidden structures and features at different levels of abstractions, and showed superior performance to LSI. However, similar to LSI, LDA, and MatchPlus, this work only utilize document-term information and does not model the relevance relationship between queries and documents. It cannot outperform the standard lexical matching based retrieval model such as *tf.idf* and BM25.

The Deep Structured Semantic Model (DSSM) proposed by Huang et al. (2013) was one of the first neural ranking models that were trained on large-scale click-through data. It represents text (query or document) by bags of letter-tri-grams; the sparse letter-tri-grams are then transformed through multiple layers of non-linear projections, generating a dense embedding for the entire query/document. The relevance between a query and a document is measured by the cosine similarity of the corresponding embeddings. The model is trained to maximize the conditional likelihood of the clicked documents given a query

using large-scale search logs. The DSSM model significantly outperformed traditional latent semantic models such as LSA, demonstrating that supervised training on user feedbacks, coupled with an optimization criterion tailored to ranking, is essential for obtaining superior document ranking performance. C-DSSM (Shen et al., 2014b) extended the DSSM architecture using convolutional neural networks to model word-n-grams. The superior performance of C-DSSM over DSSM demonstrate the effectiveness of n-grams in search ranking.

DSSM and C-DSSM build networks on top of sparse text representations rather than distributed word embeddings, which suffer from the high dimensionality of parameters. With the development of word embeddings (Mikolov et al., 2013; Pennington et al., 2014), later studies on neural ranking models shifted to using distributed word representations as input (Hu et al., 2014; Dehghani et al., 2017). The ARC-I architecture (Hu et al., 2014) takes as input the embedding of words in the sentence aligned sequentially, and summarize the meaning of a sentence through layers of convolution and pooling, until reaching a fixed length vectorial representation in the final layer; the matching of two pieces of text is done by concatenating the two vectors and feeding into a multilayer perceptron. A more recent example is the weakly supervised ranking model in which all word embeddings of a query or document are weighted-summed into one vector and the matching is done with a dense neural network (Dehghani et al., 2017).

Representation-based methods suffer from two fundamental limitations. First, the representations of the query and the document are formed without knowledge of each other; therefore they run at the risk of losing information that is important for the matching task. In traditional learning-to-rank, the most important features are the query-document features that characterize the interaction between the query and the document, which are not explicitly modeled in representation-based methods. Second, representation-based methods assume that it is possible to represent all the details of a long document with a fix-size short vector, which may not be true. A study by Guo et al. (2016b) shows that DSSM, C-DSSM, and ARC-I perform worse when trained on a whole document than when trained only on titles. Even for short text, it is well-known that controlled vocabularies, including the latent dimensions in embeddings, tend to lose detailed term matching signals which are critical to IR (Salton and McGill, 1984). Due to these limitations, most representation-based neural ranking models failed to beat unsupervised bag-of-words retrieval baselines (e.g., BM25) on academia benchmarks (Guo et al., 2016a). These drawbacks motivated the development of interaction-based neural ranking models.

Interaction-based neural ranking models explicitly model the interactions between query words and document words. Interaction-based methods are rooted in statistical translation models that estimate the probability of translating a query word to a document word (Berger and Lafferty, 1999). At query time, the ranking function considers the translation probabilities between all query-document word pairs, allowing query words to be soft-matched to document words. Traditionally, the translation between queries and documents are calculated via mutual information in documents (Berger and Lafferty, 1999; Karimzadehgan and Zhai, 2010), using document titles as queries (Jin et al., 2002) or using user clicks (Gao et al., 2010). Conventional translation models suffered from inefficiency and data sparsity because they tried to learn a probability for all the possible word pairs. With word embeddings, neural networks are able to learn and compute the translation scores in more time/data-efficient ways.

Interaction-based models consist two steps: 1) generating word-level similarity scores; and 2) combining the word-level similarity scores into a query-document relevance score. For the first step, several methods have been explored. In ARC-II (Hu et al., 2014), the model takes two word's embedding vectors and uses a convolution filter to generate the interaction score. Later studies found that it is more effective to explicitly configure the interaction function, such as using cosine similarity or dot product, than to simply concatenate the embeddings (Pang et al., 2016b,a; Guo et al., 2016a). For the second step, most of the earlier techniques treat the query-document translation matrix as an image and using 2D convolutions

to learn high level matching patterns (Hu et al., 2014; Pang et al., 2016b; Hui et al., 2017, 2018). The ARC-II (Hu et al., 2014) and MatchPyramid (Pang et al., 2016b) uses multiple layers of 2D convolutions and pooling followed by a feed-forward network. The PACRR model (Hui et al., 2017) and its extension Co-PACRR (Hui et al., 2018) employs a similar architecture but replaces the feed-forward network with a recurrent model for modeling query term dependencies. But, a study from Pang et al. (2016a) found that CNN filters tend to mix different types of match signals and thus are sub-optimal for ad-hoc retrieval. The performance of the neural ranker on an ad-hoc search task was significantly improved by separating the exact match signals from the soft match signals.

The Deep Relevance Matching Model (DRMM) (Guo et al., 2016a) introduced a new pooling technique, the histogram pooling, to combine word-word similarity scores. The histogram pooling aims to separate exact match from soft match, and to separate strong soft match from weak soft match. DRMM starts by building a translation matrix using the cosine similarity between query word and document word. Since the cosine similarity is within the interval $[-1, 1]$, this interval is discretized into a set of ordered bins, e.g. $(1, 0.8], (0.8, 0.6]...(-0.8, -1]$. DRMM then counts the number of local interactions in each bin. A separate bin $[1, 1]$ is used to count the number of exact matches between the query and the document. The histogram is used as the input to the feed-forward network to generate the matching score. DRMM also employs a term gating function to weight the contribution of each query term. The function can be frequency-based (IDF) or semantic-based (a linear function of the word embedding). DRMM demonstrated that it is more effective to separate the local interactions and count at different levels than mixing all the interaction scores using CNNs.

DRMM's results on web search (TREC Web Track) are among one of the first to demonstrate that neural ranking models can compete with classical retrieval models such as BM25 and query-likelihood model on general-domain retrieval benchmarks. However, the authors (Guo et al., 2016a) did not compare DRMM, which is a supervised model, to supervised learning-to-rank baselines. Nonetheless, DRMM demonstrates that a pure neural network based model can outperform traditional bag-of-words retrieval, which encouraged many following works on neural ranking models.

Mitra et al. (2017) showed that the interaction-based neural rankers are similar to lexical matching approaches such as BM25 and LM; representation-based neural rankers, on the other hand, are more similar to LSA. Mitra et al. (2017) combined them in a duet architecture composed of two parts: a representation-based component that learns representations of query and documents for matching in the embedding space; and an interaction-based component that builds a binary indicator matrix and extracts lexical match patterns. There are also several work focusing on modeling the internal document structures for ranking. In the NRM-F model proposed by Zamani et al. (2018b), document fields are matched to the query separately, and the matching scores are combined with a feed-forward network. The document fields include title, URL, body, as well as anchor text and previously clicked queries. Experiments show that using multiple fields performs better than a single field of the document. Fan et al. (2018) combine the match signals at different text granularities by embedding and comparing sentences and passages in addition to words. Some interaction-based models take density and position information into consideration. In DeepRank (Pang et al., 2017b), the matching is limited to text segments that contain the query words. It uses an interaction-based model similar to the MatchPyramid (Pang et al., 2016b) to match the query to each query-centric text segment, and combine the matching evidence of multiple segments using a convolution neural network. Their analysis shows that a window size around 10-20 words achieves superior performance than matching the whole text.

Starting 2018, there is a rapid progress in NLP research with the development of large pre-trained langauge models such as BERT (Lee et al., 2019a). It is pre-trained on general-purpose language modeling tasks, and the knowledge can be transferred into a variety of downstream tasks. One of the pre-training

task is to predict the relationship between two pieces of text (typically sentences), which is similar to estimating the query-document relevance. BERT uses the Transformer (Vaswani et al., 2017) architecture which can be viewed as an interaction-based neural ranking model that uses multiple attention-based layers to model the interactions of words in the first text with words in second text. This thesis dissertation (Chapter 6) and several concurrent research (Nogueira and Cho, 2019; MacAvaney et al., 2019a) show that the Transformer architecture and the large-scale language model pretraining are both effective for search ranking. BERT-based rerankers are the current state-of-the-art neural ranking models.

2.3.4 Neural Models for Initial Retrieval

Most neural IR research that we have discussed so far focused on the *reranking stage*. Recently, researchers have started to explore using neural networks to improve the *initial retrieval stage* – finding candidate documents from the entire collection. With the large scale of documents, the majority of neural ranking models discussed earlier cannot be directly applied to the initial retrieval problem due to their high complexity. The neural retrieval models must be able to improve language understanding while also retain efficiency.

One direction of research attempts to move representation-based neural ranking models to the retrieval stage. Same as representation-based neural ranking models, they use neural networks to encode a query or a document into an embedding. As the document embeddings do not rely on the queries, they can be precomputed and indexed. The potential advantage of using query or document embeddings for retrieval lies in their ability to soft match text, which can improve recall. With the embedding representations, retrieval turns into a K-nearest neighbour (KNN) search in the embedding space. When receiving a new query, after the query embedding is computed, the only operation remaining is a dot product between between the query embedding and every document embedding, which can scale to millions of candidates on a modern GPU, and billions using approximate nearest-neighbor libraries such as FAISS (Johnson et al., 2017). Note that it is less feasible to use interaction-based neural ranking models in the indexing/retrieval stage, as the ranking model needs to observe both the query and the document.

Several works have emerged along the line of developing representation-based neural retrieval models. For example, Gysel et al. (2018) proposed a neural vector space model that learns embedding representations with a shallow neural network. The model is trained in an unsupervised manner using n-gram/document pairs extracted from the corpus. Lee et al. (2019b) use the embedding-based retrieval to find candidate passages for question answering. It generates query and document embeddings using BERT (Lee et al., 2019a), retrieves a set of documents through ANN search, and use the retrieved documents to generate answers. Guu et al. (2020) improved the model from (Lee et al., 2019b) by end-to-end training the embedding-based retrieval jointly with the rest of the question-answering pipeline.

Another approach to scale KNN search is to learn sparse embeddings in which queries and documents are represented by a set of "latent words" (Salakhutdinov and Hinton, 2009; Zamani et al., 2018a). In contrast to learning low-dimensional, dense embeddings, they aim to encode queries and documents into high-dimensional but sparse embeddings. Each dimension in the embedding can be viewed as a latent word. The documents can then be stored into the inverted index where the index terms are the latent words. Zamani et al. (2018a) reported that the sparse embeddings produce more accurate rankings than its dense counterparts. An inverted index of sparse embeddings has similar speeds as a traditional bag-of-words inverted index.

All of the representation-based neural retrieval models inherit the same limitation of representation-based neural ranking models, as discussed in Section 2.3.3. They use a fixed number of dimensions, which introduces the specificity vs. exhaustiveness trade-off found in all controlled vocabularies (Salton

and McGill, 1984). While capturing high-level semantics, the latent embeddings tend to lose word-level matching information that have been fundamental to modern search engines.

Aside from learning latent embedding representations, there is another direction that modifies the discrete word-based document representations in the inverted index. Nogueira et al. (2019) proposed Doc2Query, which generates potential queries from documents using neural machine translation and indexes those queries as document expansion terms. Its effectiveness is demonstrated on a passage retrieval task, outperforming a wide range of traditional retrieval baselines. How it performs on longer documents remains to be studied. Besides Doc2Query (Nogueira et al., 2019), little attention has been paid to using neural networks to improve word-based text representation. We believe this is a research direction that worth more investigation.

2.4 Challenges in Applying Neural Networks to IR

Neural IR research has been continuing to accelerate in terms of the volume of work, the sophistication of methods, and practical effectiveness. At the same time, applying neural networks to IR still faces several challenges.

The effectiveness of neural networks for information retrieval has not been fully established. At the time this thesis research started, neural ranking models just began to show superior performance over simple bag-of-words retrieval baselines such as BM25 (Guo et al., 2016a). While a lot of complex architectures have been employed into neural ranking models, the improvements appear relatively modest when compared to traditional simple techniques. As a special type of supervised learning-to-rank method, neural ranking models were not able to outperform classic feature-based learning-to-rank systems. Among the many possible reasons, one issue that draws our attention is that the learning ability of neural networks is not fully exploited. Learning-to-rank systems are successful because they learn from a large amount of user relevance feedback. The efficacy of deep learning approaches is often driven by even larger data. Nevertheless, most existing neural ranking models only trained a shallow, small neural network. The complex network architectures were almost like manual extracting features – only that the feature extractors are coded into the neural network architectures. The key advantages of deep learning – its learning capacity – remained unexplored. To show the real power of deep learning, one should use larger data and better neural network architectures that can absorb the data.

There also remain questions about the efficiency of neural IR. Neural ranking models are relatively slow as they have more complex inputs (usually raw text content) and models (dense, distributed representations and multiple layers of networks). To compensate for speed, most of the neural ranking models are designed as a reranking system that are used to rank dozens to hundreds of documents retrieved by a faster search engine. Some of them also rely on specialized hardware (GPUs) that is relatively expensive compared to CPU architectures. Despite a few explorations (Zamani et al., 2018a; Aumüller et al., 2018; Nogueira et al., 2019), the state-of-the-art first stage retrieval is still based on exact lexical match and term-frequency based weighting, which is likely to miss many relevant documents in the first place. It is worth exploring if we can extract the patterns learned by a deep model into simple signals that can be used in a more efficient way for the first stage.

The generalization ability of neural IR models is another crucial problem. As big data is the key to unveil the true power of neural networks, it raises the question of whether neural IR models are limited to applications that have sufficient training data. For information retrieval, the most straightforward training data are the relevance feedbacks, such as human annotations and user clicks. Human annotations are expensive to obtain; user clicks takes time to collect and sometimes cause privacy concerns. How to automatically generated pseudo-relevance labels is an important research question. In addition, we question

whether the relevance patterns are the only information that neural IR models need to learn. We believe that, instead of memorizing relevance query-document pairs, a neural network should have the ability to learn knowledge from documents and provide evidence for unseen problems. Although we did not see substantial gains from the many attempts of using pre-trained word embeddings to retrieval models, more powerful pre-trained signals are still worth exploring. Finally, neural networks have turned feature engineering into neural network architecture engineering. Designing new architectures for every new problem is time-consuming. It would be of great value if the neural IR architectures can solve fundamental problems in information retrieval and thus be reusable across different search tasks.

Part II

Query-Document Interaction: Neural Ranking Models for Soft Matching

Chapter 3

K-NRM: Soft Matching Queries and Documents with Kernel-Pooling [1]

In traditional information retrieval, the ranking is based on exact lexical matches between query and document words, causing the vocabulary mismatch problem. In comparison, neural ranking models use continuous text embeddings, and models the query-document relevance via soft matches. However, prior to this thesis research, most existing neural ranking models were not able to outperform a feature-based learning-to-rank model that defined the state-of-the-art at the time.

Many neural ranking approaches use distributed representations such as word2vec (Mikolov et al., 2013) and GloVE (Pennington et al., 2014) to measure the similarity between queries and documents. Compared to the strong exact lexical match signals, soft match signals are weaker and noisier and must be used carefully. For example, word2vec may consider "dog" to be similar to "cat" and "puppy". However, a person searching for "dog" may accept a document about 'puppy', but probably will reject a document about "cat". How to use these soft match signals effectively and reliably is an open problem.

This chapter addresses these challenges with `K-NRM`, a neural ranking model with a novel kernel-pooling technique that can group soft match signals by relevance and distinguish useful soft matches from noisy ones. `K-NRM` uses word embeddings to represent query and document words. The similarities between words are pooled by a kernel-pooling layer, which uses multiple Gaussian kernels to separate different types of soft match signals. `K-NRM` then estimates the density of word pairs in different kernels, and uses the density signals as ranking features to produce a relevance score between the query and the document. All of these layers are differentiable and allow `K-NRM` to be optimized end-to-end.

The kernel-pooling layer is the key to `K-NRM`'s capability. Soft match similarities can be calculated between every possible pair of words between the query and the document, raising a challenging question on how to aggregate the many signals. In `K-NRM`'s backward pass, the kernel-pooling layer adjusts the word embeddings so that word pairs are grouped into different kernels based on their contribution to relevance. In the forward pass, it softly counts word pairs within different kernels, generating a diverse set of soft match features.

Extensive experiments on a commercial search engine's query log demonstrate the significant and robust advantages of `K-NRM`. `K-NRM` outperforms both feature-based ranking and neural ranking states-

[1] This chapter is based in full on the following previously published paper: (Xiong et al., 2017b) and (Pyreddy et al., 2018). (Xiong et al., 2017b) appears in SIGIR 2017. The proposed model, design of experiments, analysis, and paper writing were jointly shared by both authors. (Pyreddy et al., 2018) appears in SIGIR 2018. The idea of evaluating the consistency and variance was developed jointly by Dai, Xiong and Callan. The experiments wereconducted by Pyreddy, Ramaseshan and Joshi. The design of experiments, analysis, and paper writing were directed by Dai.

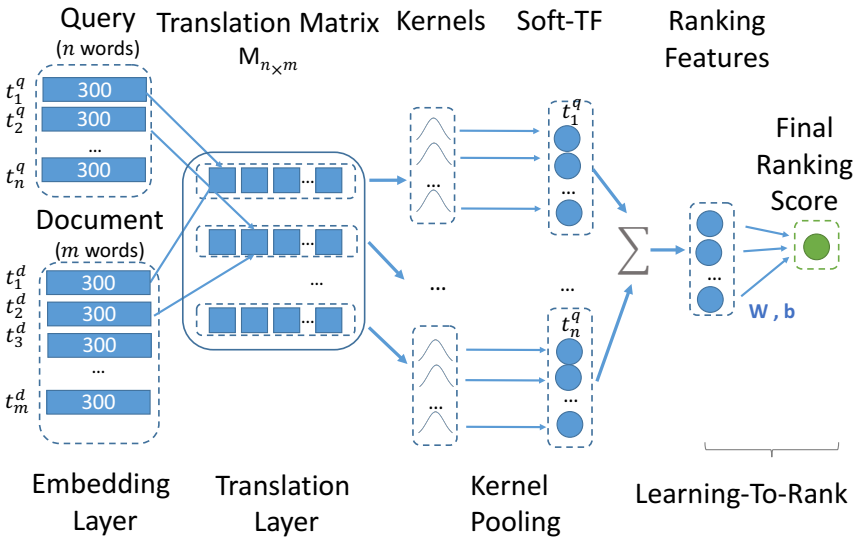


Figure 3.1: The Architecture of K-NRM. Given input query words and document words, the embedding layer maps them into distributed representations, the translation layer calculates the word-word similarities and forms the translation matrix, the kernel pooling layer generate soft-TF counts as ranking features, and the learning to rank layer combines the soft-TF to the final ranking score.

of-the-art by as much as 65%. The improvements are robust across various evaluation scenarios and different parts of the query log. Analysis show that K-NRM's advantage comes from the IR-customized multi-level soft-match patterns achieved by its kernel-guided embedding learning.

The non-convexity and stochastic training of K-NRM raises questions about its consistency compared to heuristic and learning-to-rank models that use discrete representations and simpler methods of combining evidence. We studied the stability of K-NRM, and proposed an ensemble approach to leverage the variance to build more effective and robust models.

The rest of this chapter is organized as the follows. Section 3.1 presents the kernel-based neural ranking model. Section 3.2 discusses the experimental methodologies. Section 3.3 evaluates K-NRM's effectiveness on a commercial search engine's query log. Section 3.4 studies the consistency and variance of K-NRM, and presents an ensemble approach for K-NRM. Section 3.5 concludes.

3.1 Kernel Based Neural Ranking

This section presents K-NRM, the kernel based neural ranking model. We first introduce how K-NRM produces the ranking score for a query-document pair with their words as the sole input (ranking from scratch). Then we discuss how the ranking parameters and word embeddings in K-NRM are trained from ranking labels (learning end-to-end).

3.1.1 Ranking from scratch

Given a query q and a document d, K-NRM aims to generate a ranking score $f(q, d)$ only using query words $q = \{t_1^q, ...t_i^q..., t_n^q\}$ and document words $d = \{t_1^d, ...t_j^d..., t_m^d\}$. As shown in Figure 3.1, K-NRM achieves this goal via three components: translation model, kernel-pooling, and learning to rank.

Translation Model: K-NRM first uses an embedding layer to map each word t to an L-dimension embedding $\vec{v}_t$:

$$t \Rightarrow \vec{v}_t.$$

Then a translation layer constructs a translation matrix M. Each element in M is the cosine similarity between the embeddings of a query word and a document word:

$$M_{ij} = \cos(\vec{v}_{t_i^q}, \vec{v}_{t_j^d}).$$

The translation model in K-NRM uses word embeddings to recover the word similarities instead of trying to learn one for each word pair. Doing so requires much fewer parameters to learn. For a vocabulary of size $|V|$ and the embedding dimension L, K-NRM's translation model includes $|V| \times L$ embedding parameters, much fewer than learning all pairwise similarities ($|V|^2$).

Kernel-Pooling: The translation matrix M include soft match similarities between every possible query-document word pairs, that is, $n \times m$ signals for a query with n words and a document with m words. It raises a challenging question on how to combine the many signals.

As discussed in Chapter 2, existing interaction-based neural ranking models use various ways to address this challenge. One of the most effective approaches is the Deep Relevance Matching Model (DRMM) (Guo et al., 2016a), which uses histogram pooling to assign the word-level similarities into several bins, counts the word pairs in each bin, and uses the bin sizes as ranking features. Those features let the ranker know how many match signals are strong and how many are weak, which led to state-of-the-art ranking performance at the time. However, DRMM uses a pre-trained word embedding as is, and fails when the embedding contains mixed types of word similarities, e.g., ("dog", "puppy") and ("dog", "cat"). K-NRM is inspired by DRMM's way of grouping word pairs, and aims to improve it by *learning* how to group the word pairs.

We propose a novel kernel-pooling technique. It uses density-estimation kernels with different mean values ranging from $[-1, 1]$ to assign the query-document word pairs to the corresponding similarity bins which also range from $[-1, 1]$, and softly "counts" the word pairs in each bin. Applying K kernels on the translation matrix M gives K ranking features $\phi(M)$, each corresponding to the soft term matching frequencies (Soft-TF) in one kernel:

$$\vec{K}(M_i) = \{K_1(M_i), ..., K_K(M_i)\}$$
$$\phi(M) = \sum_{i=1}^{n} \log \vec{K}(M_i)$$

$\vec{K}(M_i)$ applies K kernels to the i-th query word's row of the translation matrix. The logarithmic feature values in these bins are summed over different query terms, producing K document level ranking features.

The effect of $\vec{K}$ depends on the kernel used. This work uses the RBF kernel:

$$K_k(M_i) = \sum_j \exp(-\frac{(M_{ij} - \mu_k)^2}{2\sigma_k^2}).$$

The RBF kernel K_k calculates how word pair similarities are distributed around it: the more word pairs with similarities closer to its mean μ_k, the higher its value. μ defines the similarity level that the kernel corresponds to. A kernel with $\mu = 1$ corresponds to an exact-match term frequency feature. A kernel with $\mu = 0.5$ corresponds to the frequency of query-document word pairs that have a similarity score near 0.5. σ defines the kernel width.

Kernel pooling with RBF kernels is a generalization of existing pooling techniques. As $\sigma \to \infty$, the kernel pooling function devolves to the mean pooling. $\mu = 1$ and $\sigma \to 0$ results in a kernel that only responds to exact matches, equivalent to the term frequency values from sparse models. Otherwise, the kernel functions estimates the density of query-document word pair similarities, and produces soft-TF features [2].

Learning to Rank: The soft-TF features $\phi(M)$ are combined by a ranking layer to produce the final ranking score:

$$f(q, d) = \tanh(w^T \phi(M) + b).$$

w and b are the ranking parameters to learn. $\tanh()$ is the activation function. It controls the range of ranking score to facilitate the learning process. It is rank-equivalent to a typical linear learning to rank model.

Putting every together, `K-NRM` is defined as:

$$f(q, d) = \tanh(w^T \phi(M) + b) \qquad \text{Learning to Rank} \qquad (3.1)$$

$$\phi(M) = \sum_{i=1}^{n} \log \vec{K}(M_i) \qquad \text{Soft-TF Features} \qquad (3.2)$$

$$\vec{K}(M_i) = \{K_1(M_i), ..., K_K(M_i)\} \qquad \text{Kernel Pooling} \qquad (3.3)$$

$$K_k(M_i) = \sum_{j} \exp(-\frac{(M_{ij} - \mu_k)^2}{2\sigma_k^2}) \qquad \text{RBF Kernel} \qquad (3.4)$$

$$M_{ij} = \cos(\vec{v}_{t_i^q}, \vec{v}_{t_j^d}) \qquad \text{Translation Matrix} \qquad (3.5)$$

$$t \Rightarrow \vec{v}_t. \qquad \text{Word Embedding} \qquad (3.6)$$

Eq. 3.5-3.6 embed query words and document words, and calculate the translation matrix. The kernels (Eq. 3.4) count the soft matches between query and document's word pairs at multiple levels, and generate K soft-TF ranking features (Eq. 3.2-3.3). Eq. 3.1 is the learning to rank model.

3.1.2 Learning End-to-End

The training of `K-NRM` uses the pairwise learning to rank loss:

$$l(w, b, \mathcal{V}) = \sum_{q} \sum_{d^+, d^- \in D_q^{+,-}} max(0, 1 - f(q, d^+) + f(q, d^-)). \qquad (3.7)$$

$D_q^{+,-}$ are the pairwise preferences from the ground truth: d^+ ranks higher than d^-. The parameters to learn include the ranking parameters w, b, and the word embeddings $\mathcal{V}$.

[2] The RBF kernel is one of the most popular choices. Other kernels with similar density estimation effects can also be used, as long as they are differentiable. For example, polynomial kernel can be used, but histograms (Guo et al., 2016a) cannot as they are not differentiable.

The parameters are optimized using back propagation through the neural network. Starting from the ranking loss, the gradients are first propagated to the learning-to-rank part (Eq. 3.1) and update the ranking parameters (w, b), the kernels pass the gradients to the word similarities (Eq. 3.2-3.4), and then to the embeddings (Eq. 3.5).

The embeddings contain millions of parameters $\mathcal{V}$ and are the main capacity of the model. The learning of the embeddings is guided by the kernels. The back propagation first applies gradients from the loss function (Eq. 3.7) to the ranking score $f(q, d)$, to increase it (for d^+) or decrease it (for d^-); the gradients are propagated through Eq. 3.1 to the feature vector $\phi(M)$, and then through Eq. 3.2 to the the kernel scores $\vec{K}(M_i)$. It adjusts the corresponding kernel's score up or down to better separate the relevant document (d^+) from the irrelevant one (d^-). For this purpose, the kernels spread the gradient to word similarities in the translation matrix M_{ij}, through Eq. 3.4:

$$g(M_{ij}) = \sum_{k=1}^{K} \frac{g(K_k(M_i)) \times \sigma_k^2}{(\mu_k - M_{ij}) \exp(\frac{(M_{ij}-\mu_k)^2}{-2\sigma_k^2})}. \tag{3.8}$$

During the kernel-guided embedding learning process, a kernel pulls the word pairs closer to its μ to increase its soft-TF count, or pushes the word pairs away to reduce it, based on the gradients. The *strength* of the force also depends on the the kernel's width σ_k and the word pair's distance to μ_k: approximately, the wider the kernel is (bigger σ_k), and the closer the word pair's similarity to μ_k, the stronger the force is (Eq. 3.8). The gradient a word pair's similarity received, $g(M_{ij})$, is the combination of the forces from all K kernels.

The word embedding model receives $g(M_{ij})$ and updates the embeddings accordingly. Intuitively, the learned word embeddings are aligned by the kernels to form multi-level soft matching patterns that can separate the relevant documents from the irrelevant ones, and the learned embedding parameters $\mathcal{V}$ memorize this information. When testing, `K-NRM` extracts soft-TF features from the learned word embeddings using the kernels and produces the final ranking score using the ranking layer.

3.2 Experimental Methodology

We train and evaluate `K-NRM` on a search log from a Chinese commercial search engine, sogou.com. Training and testing labels were generated automatically from the search log using click models. We refer to this dataset as the **Sogou-Log** dataset. This section describes our experimental methods and materials on the Sogou-Log dataset.

3.2.1 The Sogou-Log Dataset

Our experiments use a query log sampled from search logs of Sogou.com, a major Chinese commercial search engine. The sample contains 35 million search sessions with $96,229$ distinct queries. The query log includes queries, displayed documents, user clicks, and dwell times. Each query has an average of 12 documents displayed. As the results come from a commercial search engine, the returned documents tend to be of very high quality.

The primary testing queries were $1,000$ queries sampled from head queries that appeared more than $1,000$ times in the query log. Most of our evaluation focuses on the head queries; we use tail query performance to evaluate model robustness. The remaining queries were used to train the neural models. Table 3.1 provides summary statistics for the training and testing portions of the search log.

The query log contains only document titles and URLs. The full texts of *testing* documents were crawled and parsed using Boilerpipe (Kohlschütter et al., 2010) for our word-based baselines (described

Table 3.1: Sogou-Log dataset characteristics.

	Training	**Testing**
Language	Chinese	
Document Fields	Title	
Queries	95,229	1,000
Documents Per Query	12.17	30.50
Search Sessions	31M	4M
Vocabulary Size	165,877	19,079

Table 3.2: K-NRM Testing Scenarios. DCTR Scores are inferred by DCTR click model (Chuklin et al., 2015). TACM Scores are inferred by TACM click model (Liu et al., 2017b). Raw Clicks use the sole click in a session as the positive label. The label distribution is the number of relevance labels from 0-4 from left to right, if applicable.

Condition	**Label**	**Label Distribution**
Testing-SAME	DCTR Scores	70%, 19.6%, 9.8%, 1.3%, 1.1%
Testing-DIFF	TACM Scores	79%, 14.7%, 4.6%, 0.9%, 0.9%
Testing-RAW	Raw Clicks	2,349,280 clicks

in Section 3.2.3). Chinese text was segmented using the open source software ICTCLAS (Zhang et al., 2003). After segmentation, documents are treated as sequences of words (as with English documents).

3.2.2 Relevance Labels and Evaluation Scenarios

Neural models like `K-NRM` and `CDSSM` (Shen et al., 2014a) require a large amount of training data. Acquiring a sufficient number of manual training labels outside of a large organization would be cost-prohibitive. User click data, on the other hand, is easy to acquire and prior research has shown that it can accurately predict manual labels. For our experiments, *training labels* were generated based on user clicks from the training sessions.

There is a large amount of prior research on building click models to model user behavior and to infer reliable relevance signals from clicks (Chuklin et al., 2015). This work uses one of the simplest click models, DCTR, to generate relevance scores from user clicks (Chuklin et al., 2015). DCTR calculates the relevance scores of a query-document pair based on their click through rates. Despite being extremely simple, it performs rather well and is a widely used baseline (Chuklin et al., 2015). Relevance scores from DCTR are then used to generate preference pairs to train our models.

The *testing labels* were also estimated from the click log, as manual relevance judgments were not made available to us. Note that there was no overlap between training queries and testing queries.

Testing-SAME infers relevance labels using DCTR, the same click model used for training. This setting evaluates the ranking model's ability to fit user preferences (click through rates).

Testing-DIFF infers relevance scores using TACM (Liu et al., 2017b), a state-of-the-art click model. TACM is a more sophisticated model and uses both clicks and dwell times. On an earlier sample of Sogou's query log, the TACM labels aligned extremely well with expert annotations: when evaluated against manual labels, TACM achieved an NDCG@5 of up to 0.97 (Liu et al., 2017b). This is substantially higher than the agreement between the manual labels generated by the authors for a sample of queries. This precision makes TACM's inferred scores a good approximation of expert labels, and Testing-DIFF is expected to produce evaluation results similar to expert labels.

Testing-RAW is the simplest click model. Following the cascade assumption (Chuklin et al., 2015), we treat the clicked document in each single-click session as a relevant document, and test whether the model can put it at the top of the ranking. Testing-Raw only uses single-click sessions (57% of the testing sessions are single-click sessions). Testing-RAW is a conservative setting that uses *raw* user feedback. It eliminates the influence of click models in testing, and evaluates the ranking model's ability to overcome possible disturbances from the click models.

The three testing scenarios are listed in Table 3.2. Following TREC methodology, the Testing-SAME and Testing-DIFF's inferred relevance scores were mapped to 5 relevance grades. Thresholds were chosen so that our relevance grades have the same distribution as TREC Web Track 2009-2012 qrels.

Search quality was measured using NDCG at depths $\{1, 3, 10\}$ for Testing-SAME and Testing-DIFF. We focused on early ranking positions that are more important for commercial search engines. Testing-RAW was evaluated by mean reciprocal rank (MRR) as there is only one relevant document per query. Evaluation used the `gdeval` evaluation software. Statistical significance was tested using the permutation test with $p < 0.05$.

3.2.3 Baselines

Our baselines include both traditional word-based retrieval models, feature-based learning-to-rank models as well as more recent neural ranking models.

Word-based baselines include `BM25` (Robertson and Walker, 1994) and language models with Dirichlet smoothing (Lm) (Lavrenko and Croft, 2001). These unsupervised retrieval methods were applied on the full text of candidate documents, and used to re-rank them. We found that these methods performed better on full text than on titles. Full text default parameters were used.

Feature-based learning-to-rank baselines include RankSVM [3], a state-of-the-art pairwise ranker, and coordinate ascent (Metzler and Croft, 2007) (Coor-Ascent[4]), a state-of-the-art listwise ranker. They use typical word-based features: Boolean AND; Boolean OR; Coordinate match; *tfidf*; BM25; language models with no smoothing, Dirichlet smoothing, JM smoothing and two-way smoothing; and bias. All features were applied to the document title and body. The parameters of the retrieval models used in feature extraction are kept default.

Neural ranking baselines include `DRMM` (Guo et al., 2016a), `CDSSM` (Shen et al., 2014a), and a simple embedding-based translation model, `Trans`.

`DRMM` was the state-of-the-art interaction based neural ranking model at the time this work was don (Guo et al., 2016a). It performs histogram pooling on the embedding based translation matrix and uses the binned soft-TF as the input to a ranking neural network. The embeddings used are pre-trained via word2vec (Mikolov et al., 2013) because the histograms are not differentiable and prohibit end-to-end learning. We implemented the best variant, $\texttt{DRMM}_{\texttt{LCH}\times\texttt{IDF}}$. The pre-trained embeddings were obtained by applying the skip-gram method from word2vec on our training corpus (document titles displayed in training sessions).

`CDSSM` (Shen et al., 2014b) is the convolutional version of `DSSM` (Huang et al., 2013). `CDSSM` maps English words to letter-tri-grams using a word-hashing technique, and uses Convolutional Neural Networks to build representations of the query and document upon the letter-tri-grams. It is a state-of-the-art representation based neural ranking model. We implemented `CDSSM` in Chinese by convolving over Chinese characters. (Chinese characters can be considered as similar to English letter-tri-grams with respect to word meaning). `CDSSM` is also an end-to-end model, but uses discrete letter-tri-grams/Chinese characters instead of word embeddings.

[3] https://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

[4] https://sourceforge.net/p/lemur/wiki/RankLib/

Table 3.3: The number of parameters and the word embeddings used by baselines and K-NRM. '–' indicates not applicable, e.g. unsupervised methods have no parameters, and word-based methods do not use embeddings.

Method	Number of Parameters	Embedding
Lm, BM25	–	–
RankSVM	21	–
Coor-Ascent	21	–
Trans	–	word2vec
DRMM	161	word2vec
CDSSM	10,877,657	end-to-end
K-NRM	49,763,110	end-to-end

Trans is an unsupervised embedding based translation model. Its translation matrix is calculated by the cosine similarity of word embeddings from the same word2vec used in DRMM, and then averaged to the query-document ranking score.

Baseline Settings: RankSVM uses a linear kernel and the hyper-parameter C was selected in the development fold of the cross validation from the range $[0.0001, 10]$. Recommended settings from RankLib were used for Coor-Ascent.

We obtained the body texts of testing documents from the new Sogou-T corpus (Luo et al., 2017b) or crawled them directly. The body texts were used by all word-based baselines. Neural ranking baselines and K-NRM used only titles for training and testing, as the coverage of Sogou-T on the training documents is low and the training documents could not be crawled given resource constraints.

For all baselines, the most optimistic choices were made: feature-based methods (RankSVM and Coor-Ascent) were trained using 10-fold cross-validation on the *testing* set and use both document title and body texts. The neural models were trained on the training set with the same settings as K-NRM, and only use document titles (they still perform better than only using the testing data). Theoretically, this gives the sparse models a slight performance advantage as their training and testing data were drawn from the same distribution.

3.2.4 K-NRM Implementation Details

The training of the K-NRM model was done on the full training data as in Table 3.1, with training labels inferred by DCTR, as described in Section 3.2.2. The embedding layer used 300 dimensions. The vocabulary size of our training data was $165,877$. The embedding layer was initialized with the word2vec trained on our training corpus. The kernel pooling layer had $K = 11$ kernels. One kernel harvests exact matches, using $\mu_0 = 1.0$ and $\sigma = 10^{-3}$. μ of the other 10 kernels is spaced evenly in the range of $[-1, 1]$, that is $\mu_1 = 0.9, \mu_2 = 0.7, ..., \mu_{10} = -0.9$. These kernels can be viewed as 10 soft-TF bins. σ is set to 0.1. The effects of varying σ are studied in Section 3.3.6.

Model optimization used the Adam optimizer, with batch size 16, learning rate $= 0.001$ and $\epsilon = 1e{-}5$. Early stopping was used with a patience of 5 epochs. We implemented our model using TensorFlow. The model training took about 50 milliseconds per batch, and converged in 12 hours on an AWS GPU machine.

Table 3.3 summarizes the number of parameters used by the baselines and K-NRM. Word2vec refers to pre-trained word embeddings using skip-gram on the training corpus. End-to-end means that the embeddings were trained together with the ranking model.Feature-based learning-to-rank models (RankSVM and Coor-Ascent) use complex, manually-crafted features, and learn simple functions to combine

those features. We found that using more complex feature transformations, e.g., kernel SVM, did not perform better. Deep neural network models, on the other hand, take raw words as features and use much larger models to transform and combine the features: CDSSM learns hundreds of convolution filters on Chinese characters, thus has millions of parameters; K-NRM's parameter space is even larger as it learns an embedding for every Chinese word. We expect that complex combinations of simple, raw-text features will fit better than shallow combinations of complex, manually-crafted features, as we will show in the next section.

3.3 Effectiveness of K-NRM

This section studies K-NRM's overall effectiveness, as well as its behavior on tail queries, with less training data, and with different kernel widths.

3.3.1 Overall Ranking Accuracy

Tables 3.4a, 3.4b and 3.5 show the ranking accuracy of K-NRM and our baselines on the Sogou-Log dataset, under three conditions.

Testing-SAME (Table 3.4a) evaluates the model's ability to fit user preferences when trained and evaluated on labels generated by the same click model (DCTR). K-NRM outperforms word-based baselines by over 65% on NDCG@1, and over 20% on NDCG@10. The improvements over neural ranking models are even bigger: On NDCG@1 the margin between K-NRM and the next best neural model is 83%, and on NDCG@10 it is 28%.

Testing-DIFF (Table 3.4b) evaluates the model's relevance matching performance by testing on TACM inferred relevance labels, a good approximation of expert labels. Because the training and testing labels were generated by different click models, Testing-DIFF challenges the model's ability to fit the underlying relevance signals despite perturbations caused by differing click model biases. Neural models with larger parameter spaces tend to be more vulnerable to this difference: CDSSM actually performs worse than DRMM, despite using thousands times more parameters. However, K-NRM demonstrates its robustness and is able to outperform all baselines by more than 40% on NDCG@1 and 10% on NDCG@10.

Testing-RAW (Table 3.5) evaluates each model's effectiveness directly by user clicks. It tests how well the model ranks the most satisfying document (the only one clicked) in each session. K-NRM improves MRR from 0.2415 (Coor-Ascent) to 0.3379. This difference is equal to moving the clicked document's from rank 4 to rank 3. The MRR and NDCG@1 improvements demonstrate K-NRM's precision oriented property—its biggest advantage is on the earliest ranking positions. This characteristic aligns with K-NRM's potential role in web search engines: as a sophisticate re-ranker, K-NRM is most possibly used at the final ranking stage, in which the first relevant document's ranking position is the most important.

The two neural ranking baselines DRMM and CDSSM perform similarly in all three testing scenarios. The *interaction* based model, DRMM, is more robust to click model biases and performs slightly better on Testing-DIFF, while the *representation* based model, CDSSM, performs slightly better on Testing-SAME.

The traditional feature-based ranking model, Coor-Ascent, performs better than all neural baselines on all three testing scenarios. The differences can be as high as 15% and some are statistically significant. This holds even for Testing-SAME which is expected to favor deep models that access more in-domain training data. These results remind that no "deep learning magic" can instantly provide significant gains for information retrieval tasks. The development of neural IR models also requires an insights

Table 3.4: K-NRM ranking accuracy on Sogou-Log Testing-SAME and Testing-DIFF. Relative performances compared with Coor-Ascent are in percentages. **W**in/**T**ie/**L**oss are the number of queries improved, unchanged, or hurt, compared to Coor-Ascent on NDCG@10. $\dagger, \ddagger, \S, \P$ indicate statistically significant improvements over Coor-Ascent†, Trans‡, DRMM§ and CDSSM¶, respectively.

(a) Testing-SAME. Testing labels are inferred by the same click model (DCTR) as the training labels used by neural models.

Method	**NDCG@1**		**NDCG@3**		**NDCG@10**		**W/T/L**
Lm	0.126	−21%	0.165	−26%	0.282	−20%	293/116/498
BM25	0.142	−11%	0.176	−22%	0.287	−10%	299/125/483
RankSVM	0.146	−9%	0.191	−15%	0.309	−13%	371/151/385
Coor-Ascent	$0.159^{\ddagger\S\P}$	–	$0.224^{\ddagger\S\P}$	–	$0.355^{\ddagger\S\P}$	–	–/–/–
Trans	0.135	−16%	0.185	−14%	0.315	−11%	318/140/449
DRMM	0.137	−14%	0.190	−15%	0.315	−11%	318/132/457
CDSSM	0.144	−10%	0.201	−10%	$0.333^{\ddagger\S}$	−6%	341/149/417
K-NRM	$\mathbf{0.264}^{\dagger\ddagger\S\P}$	+66%	$\mathbf{0.321}^{\dagger\ddagger\S\P}$	+43%	$\mathbf{0.428}^{\dagger\ddagger\S\P}$	+21%	447/153/307

(b) Testing-DIFF. Testing labels are inferred by a different click model, TACM, which approximates expert labels very well (Liu et al., 2017b).

Method	**NDCG@1**		**NDCG@3**		**NDCG@10**		**W/T/L**
Lm	0.185	−11%	0.199	−17%	0.327	−13%	369/50/513
BM25	0.163	−22%	0.189	−21%	0.325	−14%	349/53/530
RankSVM	0.170	−19%	0.204	−15%	0.352	−7%	380/75/477
Coor-Ascent	$0.2089^{\ddagger\P}$	–	$0.2403^{\ddagger}$	–	$0.3775^{\ddagger\P}$	–	–/–/–
Trans	0.187	−10%	0.213	−12%	0.345	−9%	385/68/479
DRMM	0.207	−1%	$0.249^{\ddagger}$	+4%	$0.381^{\ddagger\P}$	+1%	430/66/436
CDSSM	0.185	−11%	$0.236^{\ddagger}$	−2%	0.356	−6%	391/65/476
K-NRM	$\mathbf{0.298}^{\dagger\ddagger\S\P}$	+43%	$\mathbf{0.309}^{\dagger\ddagger\S\P}$	+28%	$\mathbf{0.420}^{\dagger\ddagger\S\P}$	+11%	474/63/395

Table 3.5: K-NRM ranking accuracy on Sogou-Log Testing-RAW. MRR evaluates the mean reciprocal rank of clicked documents in single-click sessions. Relative performance in the percentages and W(in)/T(ie)/L(oss) are compared to Coor-Ascent. $\dagger, \ddagger, \S, \P$ indicate statistically significant improvements over Coor-Ascent†, Trans‡, DRMM§ and CDSSM¶, respectively.

Method	**MRR**		**W/T/L**
Lm	0.219	−9%	416/09/511
BM25	0.228	−6%	456/07/473
RankSVM	0.224	−7%	450/78/473
Coor-Ascent	$0.242^{\ddagger}$	–	–/–/–/
Trans	0.218	−10%	406/08/522
DRMM	$0.234^{\ddagger}$	−3%	419/12/505
CDSSM	$0.232^{\ddagger}$	−4%	405/11/520
K-NRM	$\mathbf{0.338}^{\dagger\ddagger\S\P}$	+40%	507/05/424

into the advantages of neural methods and how their advantages can be incorporated to meet the needs of information retrieval tasks.

Table 3.6: Ranking performances of K-NRM variants. Relative performances and statistical significances are all compared with K-NRM's full model. $\dagger, \ddagger, \S, \P$, and $*$ indicate statistically significant improvements over K-NRM's variants of exact-match†, word2vec‡, click2vec§, max-pool¶, and mean-pool*, respectively.

	Testing-SAME				Testing-DIFF				Testing-RAW	
Variant	**NDCG@1**		**NDCG@10**		**NDCG@1**		**NDCG@10**		**MRR**	
exact-match	0.135	-49%	0.294	-31%	0.179	-40%	$0.3460^\P$	-18%	0.2147	-37%
word2vec	$0.153^\dagger$	-42%	$0.322^{\dagger\P}$	-24%	$0.216^{\dagger\P}$	-27%	$0.381^{\dagger\P}$	-10%	$0.243^{\dagger\P}$	-28%
click2vec	0.160	-39%	$0.379^{\dagger\ddagger\P}$	-11%	$0.231^{\dagger\P}$	-23%	$0.400^{\dagger\ddagger\P*}$	-4%	$0.267^{\dagger\ddagger\P}$	-21%
max-pool	0.141	-47%	0.298	-30%	0.161	-46%	0.333	-21%	0.226	-33%
mean-pool	$0.230^{\dagger\ddagger\S\P}$	-13%	$0.361^{\dagger\ddagger\S\P}$	-16%	$0.242^{\dagger\P}$	-19%	$0.379^{\dagger\P}$	-10%	$0.271^{\dagger\ddagger\P}$	-20%
full model	$\mathbf{0.264}^{\dagger\ddagger\S\P*}$		$\mathbf{0.428}^{\dagger\ddagger\S\P*}$		$\mathbf{0.298}^{\dagger\ddagger\S\P*}$		$\mathbf{0.420}^{\dagger\ddagger\P*}$		$\mathbf{0.338}^{\dagger\ddagger\S\P*}$	

3.3.2 Source of Effectiveness

K-NRM differs from previous ranking methods in several ways: multi-level soft matches, word embeddings learned directly from relevance feedback, and the kernel-guided embedding learning. This experiment studies these effects by comparing the following variants of K-NRM.

K-NRM (exact-match) only uses the exact match kernel $(\mu, \sigma) = (1, 0.001)$. It is equivalent to a basic bag-of-words retrieval model using exact match and term frequencies in the documents (*tf*).

K-NRM (word2vec) uses pre-trained word2vec, the same as DRMM. Word embedding is fixed; only the ranking part is learned.

K-NRM (click2vec) also uses pre-trained word embedding. But its word embeddings are trained on (query word, clicked title word) pairs. The embeddings are trained using skip-gram model with the same settings used to train word2vec. These embeddings are fixed during ranking.

K-NRM (max-pool) replaces kernel-pooling with max-pooling. Max-pooling finds the maximum similarities between document words and each query word; it is commonly used by neural network architectures. In our case, given the candidate documents' high quality, the maximum is almost always 1, thus it is similar to *tf*.

K-NRM (mean-pool) replaces kernel-pooling with mean-pooling. It is similar to Trans except that the embedding is trained by learning-to-rank.

All other settings are kept the same as K-NRM. Table 3.6 shows their evaluation results, together with the full model of K-NRM.

Soft match is essential. K-NRM (exact-match) performs similarly to Lm and BM25, as does K-NRM (max-pool). This is expected: without soft matches, the only signal for K-NRM to work with is effectively the *tf* score.

Ad-hoc ranking prefers relevance based word embedding. Using click2vec performs about 5-10% better than using word2vec. User clicks are expected to be a better fit as they represent user search preferences, instead of word usage in documents. The relevance-based word embedding is essential for neural models to outperform feature-based ranking. K-NRM (click2vec) consistently outperforms Coor-Ascent, but K-NRM (word2vec) does not.

Kernel-guided embedding learning provides better soft matches. K-NRM stably outperforms all of its variants. K-NRM (click2vec) uses the same ranking model, and its embeddings are trained on click contexts. K-NRM (mean-pool) also learns the word embeddings using learning-to-rank. The main difference is how the information from relevant labels is used when learning word embeddings. In

Table 3.7: Examples of word matches in different kernels. Words in **bold** are those whose similarities with the query word fall into the corresponding kernel's range (μ).

μ	Query: 'Maserati' "
1.0	**Maserati** Ghibli black interior _ who knows
0.7	Maserati **Ghibli** black interior _ who knows
0.3	Maserati Ghibli **black** interior _ **who** knows
-0.1	Maserati Ghibli black interior _ who **knows**

`KNRM (click2vec)` and `K-NRM (mean-pool)`, training signals from relevance labels are propagated *equally* to all query-document word pairs. In comparison, `K-NRM` uses kernels to enforce multi-level soft matches; query-document word pairs on different similarity levels are adjusted differently.

Table 3.7 shows an example of `K-NRM`'s learned embeddings. The **bold** words in each row are those 'activated' by the corresponding kernel: their embedding similarities to the query word 'Maserati' fall closest to the kernel's μ. The example illustrates that the kernels recover different levels of relevance matching: $\mu = 1$ is exact match; $\mu = 0.7$ matches the car model with the brand; $\mu = 0.3$ is about the car color; $\mu = -0.1$ is background noise. The `mean-pool` and `click2vec`'s uni-level training loss mix the matches at multiple levels, while the kernels provide more fine-grained training for the embeddings.

3.3.3 Kernel-Guided Word Embedding learning

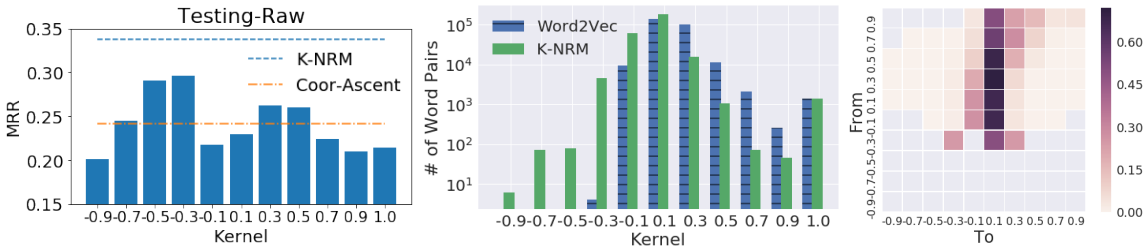
In `K-NRM`, word embeddings are initialized by `word2vec` and trained by the kernels to provide effective soft-match patterns. This experiment studies how training affects the word embeddings, showing the responses of kernels in ranking, the word similarity distributions, and the word pair movements during learning.

Figure 3.2a shows the performance of `K-NRM` when only a single kernel is used during *testing*. The x-axis is the μ of the kernel. The results indicate the kernels' importance. The kernels on the far left (≤ -0.7), the far right (≥ 0.7), and in the middle ($\{-0.1, 0.1\}$) contribute little; the kernels on the middle left ($\{-0.3, -0.5\}$) contribute the most, followed by those on the middle right ($\{0.3, 0.5\}$). Higher μ does not necessarily mean higher importance or better soft matching. Each kernel focuses on a group of word pairs specific soft match relationships; the importance of this group is learned by the model.

Figure 3.2b shows the number of word pairs activated in each kernel before training (`Word2Vec`) and after (`K-NRM`). The X-axis is the kernel's μ. The Y-axis is the log number of word pairs activated (whose similarities are closest to corresponding kernel's μ). Most similarities fall into the range (-0.4, 0.6). These histograms help explain why the kernels on the far right and far left do not contribute much: because there are fewer word pairs in them.

Figure 3.2c shows the word movements during the embedding learning. Each cell in the matrix contains the word pairs whose similarities are moved from the kernel in the corresponding row (μ on the left) to the kernel in the corresponding column (μ at the bottom). The color indicates the fraction of the moved word pairs in the original kernel. Darker indicates a higher fraction. Several examples of word movements are listed in Table 3.8. Combining Figure 3.2 and Table 3.8, the following trends can be observed in the kernel-guided embedding learning process.

Many word pairs are decoupled. Most of the word movements are from other kernels to the "white noise" kernels $\mu \in \{-0.1, 0.1\}$. These word pairs are considered related by `word2vec` but not by `K-NRM`. This is the most frequent effect in `K-NRM`'s embedding learning. Only about 10% of word pairs with similarities ≥ 0.5 are kept. This implies that document ranking requires a stricter measure of soft



(a) Individual kernel's performance (b) Kernel sizes (c) Word pair movements

Figure 3.2: Kernel guided word embedding learning in K-NRM. Fig. 3.2a shows the performance of K-NRM when only one kernel is used in *testing*. Its X-axis is the μ of the used kernel. Its Y-axis is the MRR results. Fig. 3.2b shows the log number of word pairs that are closest to each kernel, before K-NRM learning (Word2Vec) and after. Its X-axis is the μ of kernels. Fig. 3.2c illustrates the word pairs' movements in K-NRM's learning. The heat map shows the fraction of word pairs from the row kernel (before learning, μ marked on the left) to the column kernel (after learning, μ at the bottom).

match. For example, as shown in Table 3.8's first row, a person searching for 'China-Unicom', one of the major mobile carriers in China, is less likely interested in a document about 'China-Mobile', another carrier; in the second row, 'Maserati' and 'car' are decoupled as 'car' appears in almost all candidate documents' titles, so it does not provide much evidence for ranking.

New soft match patterns are discovered. K-NRM moved some word pairs from near zero similarities to important kernels. As shown in the third and fourth rows of Table 3.8, there are word pairs that less frequently appear in the same surrounding context, but convey possible search tasks, for example, "the *search* for *MH370* ". K-NRM also discovers word pairs that convey strong "irrelevant" signals, for example, people searching for "BMW" are not interested in the "contact us" page.

Different levels of soft matches are enforced. Some word pairs moved from one important kernel to another. This may reflect the different levels of soft matches K-NRM learned. Some examples are in the last two rows in Table 3.8. The -0.3 kernel is the most important one, and received word pairs that encode search tasks; the 0.5 kernel received synonyms, which are useful but not the most important, as exact match is not that important in our setting.

3.3.4 Required Training Data Size

This experiment studies K-NRM's performance with varying amounts of training data. Results are shown in Figure 3.3. The X-axis is the number of sessions used for training (e.g. $8K, 32K, \ldots$), and the coverage of testing vocabulary in the learned embedding (percentages). Sessions were randomly sampled from the training set. The Y-axis is the performance of the corresponding model. The straight and dotted lines are the performances of Coor-Ascent.

When only 2K training sessions are available, K-NRM performs worse than Coor-Ascent. Its word embeddings are mostly unchanged from word2vec as only 16% of the testing vocabulary are covered by the training sessions. K-NRM's accuracy grows rapidly with more training sessions. With only 32K (0.1%) training sessions and 50% coverage of the testing vocabulary, K-NRM surpasses Coor-Ascent on Testing-RAW. With 128K (0.4%) training sessions and 69% coverage on the testing vocabularies, K-NRM surpasses Coor-Ascent on Testing-SAME and Testing-DIFF. The increasing trends against Testing-SAME and Testing-RAW have not yet plateaued even with 31M training sessions, suggesting that K-NRM can utilize more training data. The performance on Testing-DIFF plateaus after 500K sessions,

Table 3.8: Examples of moved word pairs in K-NRM. **From** and **To** are the μ of the kernels the word pairs were in before learning (word2vec) and after (K-NRM). Values in parenthesis are the individual kernel's MRR on Testing-RAW, indicating the importance of the kernel. '+' and '−' mark the sign of the kernel weight w_k in the ranking layer; '+' means word pair appearances in the corresponding kernel are positively correlated with relevance; '−' means negatively correlated.

From	To	Word Pairs
$\mu = 0.9$ (0.20, −)	$\mu = 0.1$ (0.23, −)	(wife, husband), (son, daughter), (China-Unicom, China-Mobile)
$\mu = 0.5$ (0.26, −)	$\mu = 0.1$ (0.23, −)	(Maserati, car),(first, time) (website, homepage)
$\mu = 0.1$ (0.23, −)	$\mu = -0.3$ (0.30, +)	(MH370, search), (pdf, reader) (192.168.0.1, router)
$\mu = 0.1$ (0.23, −)	$\mu = 0.3$ (0.26, −)	(BMW, contact-us), (Win7, Ghost-XP)
$\mu = 0.5$ (0.26, −)	$\mu = -0.3$ (0.30, +)	(MH370, truth), (cloud, share) (HongKong, horse-racing)
$\mu = -0.3$ (0.30, +)	$\mu = 0.5$ (0.26, −)	(oppor9, OPPOR), (6080, 6080YY), (10086, www.10086.com)

perhaps because the click models do not perfectly align with each other; more regularization of the K-NRM model might help under this condition.

3.3.5 Performance on Tail Queries

This experiment studies how K-NRM performs on less frequent queries. We split the queries in the query log into Tail (less than 50 appearances), Torso (50-1000 appearances), and Head (more than 1000 appearances). For each category, 1000 queries are randomly sampled as testing; the remaining queries are used for training. Following the same experimental settings, the ranking accuracies of K-NRM and Coor-Ascent are evaluated.

The results are shown in Table 3.9. Evaluation is only done using Testing-RAW as the tail queries do not provide enough clicks for DCTR and TACM to infer reliable relevance scores. The results show an expected decline of K-NRM's performance on rarer queries. K-NRM uses word embeddings to encode the relevance signals. As tail queries' words appear less frequently in the training data, it is hard to generalize the embedded relevance signals through them. Nevertheless, even on queries that appear less than 50 times, K-NRM still outperforms Coor-Ascent by 8%.

3.3.6 Hyper Parameter Study

This experiment studies the influence of the kernel width (σ). We varied the σ used in K-NRM's kernels, kept everything else unchanged, and evaluated its performance. The shapes of the kernels with 5 different σ and the corresponding ranking accuracies are shown in Figure 3.4. Only Testing-RAW is shown due to limited space; the observation is the same on Testing-SAME and Testing-DIFF.

As shown in Figure 3.4, kernels too sharp or too flat either do not cover the similarity space well, or mixed the matches at different levels; they cannot provide reliable improvements. With σ between 0.05 and 0.2, K-NRM's improvements are stable.

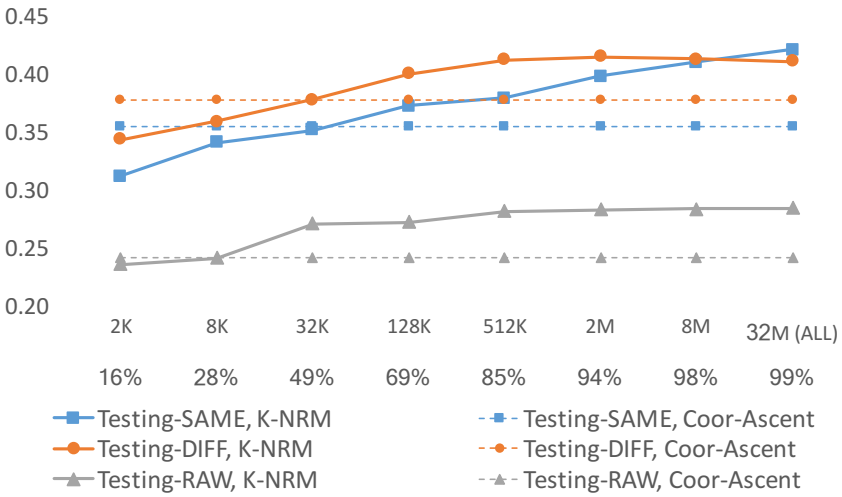


Figure 3.3: K-NRM's performances with different amounts of training data. X-axis: Number of sessions used for training, and the percentages of testing vocabulary covered (second row). Y-axis: NDCG@10 for Testing-SAME and Testing-DIFF, and MRR for Testing-RAW.

Table 3.9: Ranking accuracy on Tail (frequency < 50), Torso (frequency $50 - 1K$) and Head (frequency $> 1K$) queries. † indicates statistically significant improvements of K-NRM over Coor-Ascent on Testing-RAW. **Frac** is the fraction of the corresponding queries in the search traffic. **Cov** is the fraction of testing query words covered by the training data.

	Frac	**Cov**	**Testing-RAW, MRR**		
			Coor-Ascent	K-NRM	
Tail	52%	85%	0.2977	0.3230†	+8.49%
Torso	20%	91%	0.3202	0.3768†	+17.68%
Head	28%	99%	0.2415	0.3379†	+39.92%

We have also experimented with several other structures for K-NRM, for example, using more learning to rank layers, and using *idf* to weight query words when combining their kernel-pooling results. However we have only observed similar or worse performances. Thus, we chose to present the simplest successful model to better illustrate the source of its effectiveness.

Summary: the experiments in Section 3.3 demonstrates the effectiveness of K-NRM. On three testing scenarios, K-NRM outperforms both feature based ranking baselines and neural ranking baselines by large margins, and is extremely effective at the top ranking positions. Our analysis revealed that the advantage of K-NRM is IR-customized multi-level soft match between query and documents, which is achieved by the novel kernel-guided embedding learning.

3.4 Consistency and Variation of K-NRM

The non-convexity and stochastic training of K-NRM raises questions about its consistency compared to heuristic and learning-to-rank models that use discrete representations and simpler methods of combining evidence. Consistent behavior under slightly different conditions is essential to reproducible research and deployment in industry.

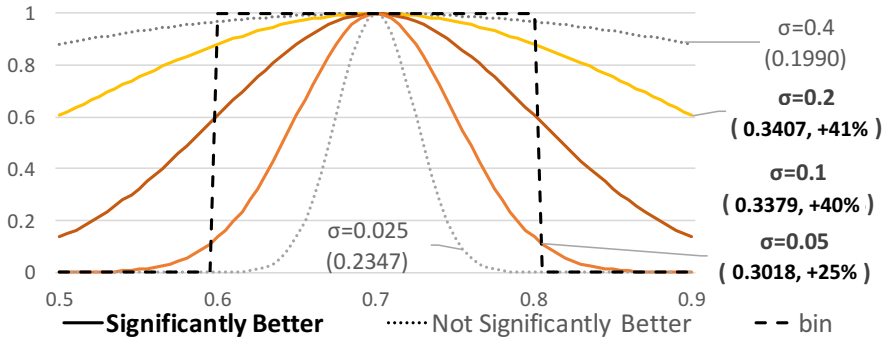


Figure 3.4: K-NRM's performance with different σ. MRR and relative gains over Coor-Ascent are shown in parenthesis. Kernels drawn in solid lines indicate statistically significant improvements over Coor-Ascent.

This section studies the stability of K-NRM. It compares the behavior of multiple trained models. We find that although K-NRM produces similar accuracy across different trials, it also produces rather different document rankings for individual queries. It further investigates the learned patterns across the different trials to better understand the variance. The difference in the ranking patterns from different K-NRM trials makes them a good fit for ensembles. An ensemble approach is proposed to leverage the variance to build more robust and generalizable models.

This section starts with a briefly discussion on the experimental methodology. Next, it compares the ranking accuracy of multiple trained models, and investigates the learned patterns in these models. Then, it presents results on our ensemble approach.

3.4.1 Experimental Methodology

To investigate the consistency and variation of K-NRM, 50 statically trained models are generated with random parameter initialization. Due to the non-convexity of the optimization problem, the 50 trials would converge differently.

Model implementation, training, and testing all followed the methodology defined on the **Sogou-Log** task (Section 3.3). Training and testing labels are generated from click models. Three testing conditions, Testing-SAME, Testing-DIFF ,and Testing-RAW, were used; Testing-DIFF and Testing-RAW were considered more reliable than Testing-SAME because they are less subject to over-fitting. Different from the original methodology the previous Sogou-Log evaluation(Section 3.3), we built the vocabulary from the queries, titles and URLs for better term coverage; the original setup used only the queries and titles. Through our experiments we observed that the loss saturates after 2 epochs on the validation set, hence we used that as the stopping condition in all experiments.

3.4.2 Variance

The first experiment studied the consistency of K-NRM by running 50 stochastically trained models with random initialization. The consistency among the 50 trials is examined at the query-set level and the individual query level.

The performance of the 50 models on the three metrics is summarized in Table 3.10. The min/max differences can be large, especially for NDCG@1. However the standard deviations are small, ranging in 0.5-1.3% absolute, and 1-4% relative to mean values. We identified that the min/max differences are due

Table 3.10: Statistics from 50 K-NRM trials trained with random parameter initialization. All trials were configured and evaluated in exactly the same procedure. Minimum, Mean, and Maximum are the worst, average, and best performances. Standard Deviation is calculated on the corresponding 50 evaluation scores.

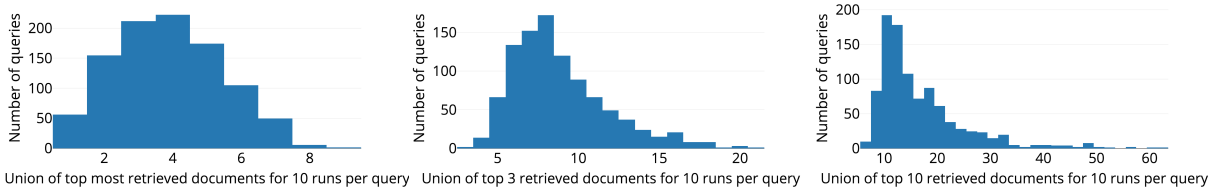
	Testing-SAME		Testing-DIFF		Testing-RAW
Model	**NDCG@1**	**NDCG@10**	**NDCG@1**	**NDCG@10**	**MRR**
Minimum	0.253	0.422	0.298	0.426	0.343
Mean	0.286	0.440	0.324	0.438	0.355
Maximum	0.317	0.458	0.348	0.450	0.370
Standard Deviation	0.013	0.010	0.011	0.005	0.007
Results in Sec. 3.3	0.264	0.428	0.298	0.420	0.338

to a small number of outliers and performance is stable across most trials. We also show the results in the previous Sogou-Log evaluation (Setion 3.3) in Table 1. The old results fall into the lower end of our trials due to different vocabularies and stopping conditions.

The next analysis studied the consistency at the individual query level by examining document rankings generated by different trials. For each query we examined the top k ranked document from 10 different trials. The total number of distinct documents indicates how well the models agree about which documents to place at the top of the ranking. A histogram (Figure 3.5) shows the number of queries at each agreement level for top 1, 3, and 10 ranked documents.

Different trials rank different documents at the top to a certain extent. For about 50 of the 1000 queries, all 10 trials select the same document at rank 1 (Figure 3.5a); for 35% of the queries, the trials select 2-3 different documents. Moderate consistency is observed across the trials. Only 15% of the queries get more than 5 different documents from the 10 trials. None of the queries get 10 completely different documents at the top1, which means that for every query at least 2 trials agree with each other on the most relevant document.

A similar trend is seen in Figure 3.5b, where for 66% of the queries, the 10 trials collectively select 3-9 different documents to fill the top 3 slots. The document sets from the 10 trials converge deeper in the rankings. In the top 10 rankings (Figure 3.5c), the histogram shifts to the left, indicating that the 10 trials have higher agreement. This is expected because K-NRM only re-ranks the top 30 documents. The disagreement in the top 1 and 3 ranks indicates that even though different trials largely have the same sets of documents, their rankings are slightly different.



(a) Agreement on Top 1 (b) Agreement on Top 3 (c) Agreement on Top 10

Figure 3.5: Query level ranking agreement of K-NRM. The X-axes are the number of distinct documents that appeared in the top K ranking results of 10 K-NRM trials. Larger X values indicate lower agreement among trials. Y-axes are the number of queries with rankings at the corresponding agreement level.

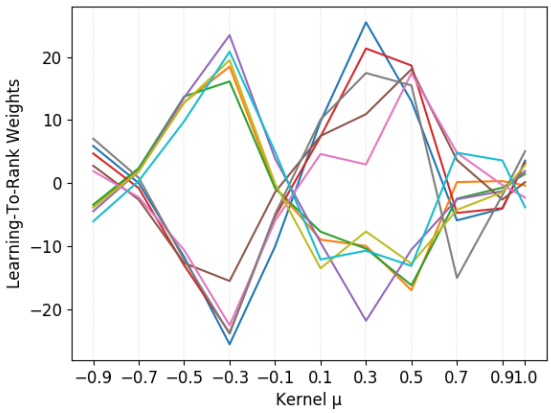
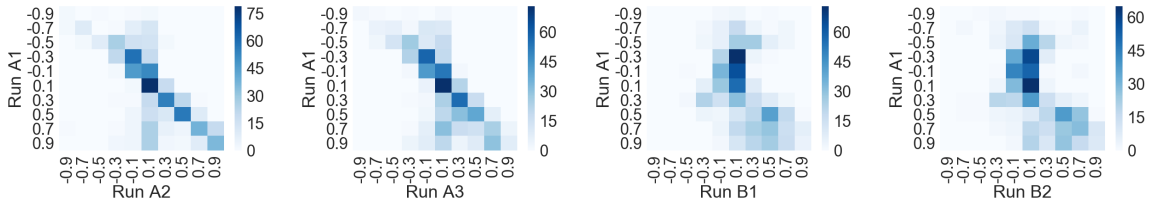


Figure 3.6: Learning to rank weights from 10 K−NRM trials. The X axis is the μ of a kernel. The Y axis is its ranking weight.



(a) Run A1 vs. Run A2 (b) Run A1 vs. Run A3 (c) Run A1 vs. Run B1 (d) Run A1 vs. Run B2

Figure 3.7: Word pair movements between runs from two patterns of K−NRM. Pattern A includes Runs A1, A2, and A3. Pattern B includes B1 and B2. Each cell (μ_x, μ_y) in the heat map indicates the fraction of word pairs whose cosine similarities fall into Kernel μ_y in Run A1 (Y-axis) and kernel μ_x in the other run (X-axis). Darker cell indicates more word pairs.

3.4.3 Latent Matching Patterns

To better understand the model differences, we investigated the model parameters through multiple K−NRM trials. K−NRM has two trainable components: the word embedding layer and the learning-to-rank layer. The word embedding layer aligns query-document word pairs and assigns them to the closest kernels by their cosine similarity. The learning-to-rank layer learns the importance of word pairs around each kernel. This analysis studied both parameters.

Figure 3.6 plots the learning-to-rank weights from 10 random trials. The trials fall into two main patterns. One pattern starts with a downward slope and then moves upward while the second pattern goes the other way. K−NRM allocates word pairs into corresponding kernels based on their contribution to relevance. Different learning-to-rank weights indicate different ways of allocating word pairs to kernels.

We further studied the two patterns with word embeddings from multiple trials. We randomly picked 5 runs. Runs A1–A3 belonged to one learning-to-rank weight pattern (Pattern A); runs B1–B2 belonged to the other pattern (Pattern B). We compared the word pair distribution between pairs of runs through a heat map with each cell (μ_x, μ_y) indicating the fraction of word pairs that fall into kernel μ_x in one run and kernel μ_y in the other run. Figure 3.7 shows the heat maps between Run A1 and the rest of runs.

Table 3.11: The performance of K-NRM ensemble models using different base models. K-NRM Mean is the average performance of 50 base models. $*, \dagger, \S$ indicate statistically significant improvements over K-NRM Mean, Ensemble-A and Ensemble-B respectively. Statistical significance was tested using the permutation test with $p < 0.05$.

	Testing-SAME		Testing-DIFF		Testing-RAW
Model	**NDCG@1**	**NDCG@10**	**NDCG@1**	**NDCG@10**	**MRR**
Results in Sec. 3.3	0.264 (-7%)	0.428 (-3%)	0.298 (-8%)	0.420 (-4%)	0.338 (-5%)
K-NRM Mean	0.286	0.440	0.324	0.438	0.355
Ensemble-A	0.327 (14%)$*$	0.469 (7%)$*$	0.370 (14%)$*$	0.457 (4%)$*$	0.391 (10%)$*$
Ensemble-B	0.322 (12%)$*$	0.457 (4%)$*$	0.383 (18%)$*$	0.463 (6%)$*$	0.393 (11%)$*$
Ensemble-A&B	0.336 (17%)$*\dagger\S$	0.469 (7%)$*\dagger\S$	0.393 (21%)$*\dagger\S$	0.468 (7%)$*\dagger\S$	0.404 (14%)$*\dagger\S$

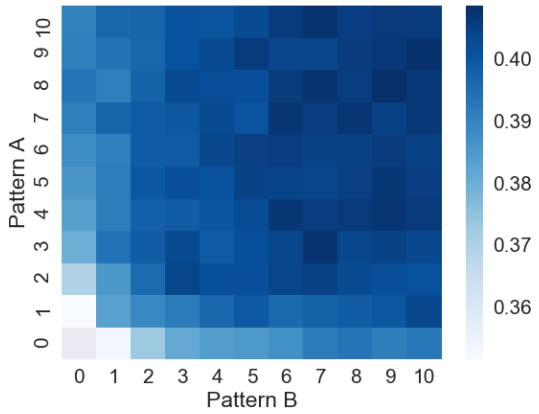


Figure 3.8: The accuracy of ensemble models that combine different numbers of base models from Patterns A and B. Each cell is the MRR (Testing-RAW) of an ensemble model built with m Pattern A models (Y-axis) and n Pattern B models (X-axis). Darker color indicates higher accuracy.

Runs from the same pattern have similar heat maps. As shown in Figure 3.7a and 3.7b, Runs A2 and A3 show a strong diagonal pattern, indicating that most of the word pairs are in the same kernel as in run A1. Runs from pattern B share another word pair distribution. As can be seen from Figure 3.7c and 3.7d, A lot of word pairs are assigned to a different kernel by runs B1/B2 as compared to the kernel assigned by run A1. The results reveal two distinct latent matching patterns. Trials from the same pattern have similar learning-to-rank weights and word embeddings. The two patterns differ largely in their word pair alignment.

Although the two patterns align word embeddings differently, both are equally effective and produce similar accuracy (Table 1).

3.4.4 Ensemble Model

The different rankings and distinct patterns in multiple K-NRM trials provided possibilities to reduce risk and improve the model's generalization ability using ensemble models (Krogh and Vedelsby, 1994). The following experiments studied the effectiveness of ensemble models.

We used an unweighted-average ensemble model (Krizhevsky et al., 2012) that averages the scores from multiple trials. To investigate the effects of latent matching patterns, we tested different types of ensemble models: `Ensemble-A` used 10 base models randomly selected from Pattern A; `Ensemble-B` used 10 base models from Pattern B; `Ensemble-A&B` used base models from both patterns, 5 from each[5]. To make evaluation reliable, 10 ensemble rankers were generated for each method with different base models randomly chosen from a pool of 50 `K-NRM` trials.

All ensemble methods significantly outperformed individual models (Table 3.11). The differences in document rankings allowed multiple trials to 'vote' in the ensemble model. Documents favored by the majority of trials are voted up, whereas documents that are mistakenly ranked highly in a poor trial are voted down. This reduces the risk and improves the accuracy. Comparing NDCG scores at different depths, we see that ensembles are most effective at the top of the ranking. This is because the dataset mostly contains 20-30 documents per query. There is more opportunity for disagreement at the top, which gives more scope for improvement. Ensembles also improved model generalization. The generalization ability is reflected by Testing-DIFF and Testing-RAW whose relevance labels are different from the labels used in training. As can be seen from in Table 2, the performance on the two test sets were both improved by large margins; Testing-DIFF gains more than Testing-SAME.

`Ensemble-A&B` outperformed `Ensemble-A` and `Ensemble-B` (Table 2), which indicates that having and *recognizing* two distinctive matching patterns is beneficial to ensemble models.

To further understand the effects of the two patterns, we tested ensemble models with `m` Pattern A models and `n` Pattern B models. Figure 3.8 shows MRR on Testing-RAW as a heatmap. It confirms that having two variations enables better ensembles; ensemble models that only used one pattern have the lowest accuracy. Comparing to single pattern ensembles, mixed ensembles can achieve the same accuracy using a smaller ensemble model with fewer base models. For example, cell $(3, 3)$ has higher accuracy than cell $(10, 0)$. Besides, ensemble models benefit from a balanced mix of the two patterns, as seen from the darker cells around the diagonal which have similar number of base trials from each pattern. Prior research did not recognize that `K-NRM` consistently converges to a small number of distinct, equally-good local optima. Recognizing this helps in constructing high-quality ensembles.

Summary: the experiments in Section 3.4 find that the accuracy of `K-NRM` is quite stable (has low standard deviation) in spite of its random components. However, stable NDCG does not imply identical rankings at the individual query level. Multiple trials of `K-NRM` converge to two latent patterns that are about equally effective; runs within the same pattern converge to similar ranking weights and word embeddings. This behavior was not recognized by prior work, and is worth additional study. Finally, the distinct but equally effective matching patterns makes `K-NRM` a good fit for ensemble, which further improves `K-NRM`'s accuracy and ability to generalize.

3.5 K-NRM Summary

This chapter presents `K-NRM`, a kernel based neural ranking model to bring soft match into bag-of-words ranking models. It uses multiple Gaussian kernels to softly count soft matches at different similarity levels. Supervised by ranking labels, it learns to adjust the word embeddings so that the soft match signals are grouped based on their contribution to relevance.

Experiments on the Sogou-Log dataset from a commercial search engine demonstrated the advantage of `K-NRM`. On three testing scenarios, `K-NRM` outperforms both feature based ranking baselines and neural ranking baselines by large margins, and is extremely effective at the top ranking positions. The improvements are also robust: Stable gains are observed on head and tail queries, with fewer training

[5] We found that performance saturates with more than 10 base models.

data, a wide range of kernel widths, and a simple ranking layer. This work was the state-of-the-art at the time.

Our analysis revealed that the advantage of K-NRM is the the IR-customized multi-level soft-TF signals between query and documents, achieved with the kernel-guided embedding learning. Without it, K-NRM's advantage quickly diminishes: its variants with only exact match, pre-trained word2vec, or uni-level embedding training all perform significantly worse, and sometimes fail to outperform the simple feature based baselines. By grouping and counting soft matched words, K-NRM also provides certain levels of interpretability: analyzing the word pairs in each kernel, we found that more than 90% of word pairs that are mapped together in word2vec are decoupled, satisfying the stricter definition of soft match required in ad-hoc search. Word pairs that are less correlated in documents but convey frequent search tasks are discovered and mapped to certain similarity levels. The kernels also moved word pairs from one kernel to another based on their different roles in the learned soft match.

By investigating multiple trials of K-NRM, we found that the accuracy of K-NRM is quite stable (has low standard deviation) in spite of its random components. However, stable NDCG does not imply identical rankings at the individual query level. Different trials have moderate agreement about which document to rank first. Ten trials collectively select 1-3 documents to rank first for 40% of our queries. Multiple trials of K-NRM converge to two latent patterns that are about equally effective. Runs within the same pattern converge to similar ranking weights and word embeddings. The distinct but equally effective matching patterns makes K-NRM a good fit for ensemble models. Recognizing different convergence patterns and selecting ensemble components equally from each pattern further improves K-NRM's accuracy and ability to generalize.

To the best of our knowledge, K-NRM was the first neural ranking model to outperform a learning-to-rank system that uses exact lexical match feature, proving the effectiveness of neural networks for ranking problems. The kernel-pooling techique provides an effective and easy-to-interpret approach to learning, grouping, and counting soft matches. This technique may improve the effectiveness and interpretability of a wide range of text matching problems, e.g., question answering and natural language inference. For information retrieval, the kernel-pooling technique also has more potential applications than described in this chapter. In the next chapter, we will show how kernel-pooling can be used to soft match n-grams in information retrieval.

Chapter 4

Conv-KNRM: Soft Matching N-grams with Convolutional Neural Networks[1]

The previous chapter demonstrates the effectiveness of using neural networks to learn soft match patterns between *words*. On the other hand, the query and document often match at *n-grams*, such as phrases, concepts and entities. This chapter studies how to effectively model n-gram soft match patterns to improve relevance ranking.

There has been a large amount of research that utilizes n-gram exact matches. For example, the sequential dependency model (SDM) that includes n-gram phrase matches has been a standard in many IR systems (Metzler and Croft, 2005) . There is also some work that uses entities to introduce explicit semantics from knowledge graphs to search systems (Xiong et al., 2016). The majority of these studies treat n-grams as discrete terms and use them the same as unigrams. For example, a document bigram 'white_house' is one term, has its own term frequency, and can only be matched to 'white_house' in queries. The discrete representation makes it infeasible to model the soft matching between n-grams – the model needs to learn the correlation between every possible n-gram pair, which inevitably faces data sparsity and parameter explosion.

Modeling n-grams is much easier in the embedding space. Neural methods have shown the benefits of modeling n-grams in some related text processing tasks, especially with Convolutional Neural Networks (CNN). For example, in sentence classification, CNN has been used to compose word embeddings into n-gram representations, which are then max-pooled and combined by a feed-forward neural network to classify the sentence (Kim, 2014). That research demonstrated CNN's ability of composing n-gram embeddings, while its ability in IR is still being explored.

This chapter aims to leverage CNN to model n-gram soft match for relevance ranking. It presents `Conv-KNRM`, a convolutional kernel-based neural network for soft matching n-grams. It first embed words in continuous vectors (embeddings), and then employ Convolutional Neural Networks (CNN) to compose n-gram embeddings from adjacent words' embeddings. In the n-gram embedding space, it applies the kernel-pooling technique from `K-NRM` (Chapter 3) to combine n-gram soft match signals to the final ranking score. The CNN is the key to modeling n-grams. It avoids the curse of dimensionaltiy problem by learning a convolutional layer that forms n-grams from individual words' embeddings. The convolutional layer projects all n-grams into a unified embedding space, allowing matching n-grams of

[1]This chapter is based in full on a previously published paper (Dai et al., 2018) appearing in WSDM 2018. The model framework was implemented and evaluated by Dai. The idea of using CNNs for modeling n-gram was originated from Xiong. The design of experiments, analysis, and paper writing were jointly shared by both authors.

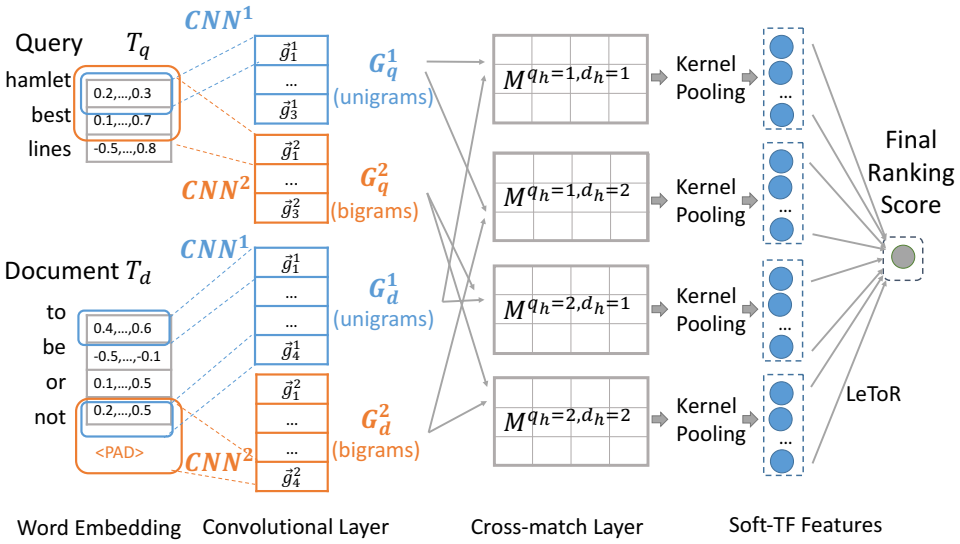


Figure 4.1: The Conv-KNRM Architecture. Given input query and document, the embedding layer maps their words into distributed representations; the convolutional layer generates n-gram embeddings; the cross-match layer matches the query n-grams and document n-grams of different lengths, and forms the translation matrices; the kernel pooling layer generates soft-TF features and the learning-to-rank (LeToR) layer combines them to the ranking score. The case with unigrams and bigrams ($h_{max} = 2$) is shown. In experiments we used uptp $h_{max} = 3$.

different lengths. For instance, 'white house' in the document can provide partial evidence for the query 'George Walker Bush'.

The whole Conv-KNRM model can be trained end-to-end with relevance signals such as clicks, so that the n-gram soft matches are fully optimized towards search accuracy. This chapter also explores the domain adaptation ability of Conv-KNRM. It presents a simple yet effective domain adaptation method for applying Conv-KNRM to search domains where large scale training data is not available. We first train the word embedding and convolutional layers in the source domain that has sufficient training labels. The trained Conv-KNRM is then adapted to a target domain with limited annotations by only re-training the learning-to-rank layer. The assumption is that the soft matching patterns learned on one domain are likely to generalize to similar domains, while the importance of each type of soft match can vary across domains.

The rest of this chapter is organized as the follows. Section 4.1 describes the model architecture of Conv-KNRM; Section 4.2 describes the domain adaptation method; Experimental setups and evaluation results are presented in Section 4.3 and Section 4.4. Section 4.5 summarizes Conv-KNRM.

4.1 Convolutional N-Gram Ranking

This section presents the convolutional kernel-based neural ranking Model (Conv-KNRM), shown in Figure 4.1. It first composes n-gram embeddings using CNN, and constructs translation matrices between n-grams of different lengths in the n-gram embedding space). Then it ranks with the n-gram soft matches using kernel-pooling and learning to rank.

4.1.1 N-gram Composing and Cross-Matching

Given a query q and document d, `Conv-KNRM` embeds their words by a word embedding layer, composes n-grams with a CNN layer, and cross-matches query n-grams and document n-grams of variant lengths to the translation matrices.

Embedding layer: `Conv-KNRM` first maps each word t of a text to an L-dimensional continuous vector (embedding): $t \rightarrow \vec{t}$. A query q or document d is treated as a text sequence of m words $\{t_1, ..., t_m\}$, and is modeled as an $m \times L$ matrix:

$$T = \begin{bmatrix} \vec{t}_1 \\ ... \\ \vec{t}_m \end{bmatrix}. \tag{4.1}$$

We denote the embedding matrix of the query and the document by T_q and T_d respectively.

Convolutional Layer: `Conv-KNRM` than applies convolution filters to compose n-grams from the text (T_q or T_d). A convolution filter slides over the text like a sliding window, generating n-grams on-the-fly. For each window of h words, the filter sums up all elements in the h words' embeddings $T_{i:i+h}$, weighted by the filter weights $w \in \mathbb{R}^{hL}$,and produces a continuous score:

$$v = w \cdot T_{i:i+h}, v \in \mathbb{R}. \tag{4.2}$$

Using F different filters $w_1, ..., w_F$ gives F scores, each describing $T_{i:i+h}$ in a different perspective. Then we add a bias and apply a non-linear activation function, and obtain an F-dimensional embedding for the h-gram:

$$\vec{g}_i^h = \text{relu}\left(W^h \cdot T_{i:i+h} + \vec{b}^h\right), i = 1...m. \tag{4.3}$$

$\vec{g}_i^h \in \mathbb{R}^F$ is the embedding of the i-th h-gram. The f-th element in $\vec{g}_i^h$ is the score of the f-th filter. W^h and $\vec{b}^h$ are the weights of the F convolution filters. $|W^h| = (hL) \times F$ and $|\vec{b}^h| = F$. When a convolution filter slides across the boundary of the text, we append up to $h-1$ '<PAD>' symbols for padding.

Thus, for each n-gram length $h \in \{1, .., h_{max}\}$, the CNN layer converts the text embedding T into h-gram embedding G^h.

$$G^h = \text{CNN}^h(T) = \begin{bmatrix} \vec{g}_1^h \\ ... \\ \vec{g}_m^h \end{bmatrix} \tag{4.4}$$

$|G^h| = m \times F$. Each of its rows correspond to a h-gram vector of length F. h-gram embeddings for the query and the document are denoted as G_q^h and G_d^h respectively.

The "convolution" assumption is applied in the n-gram compositions: the same set of convolution filters is used to compose all n-grams, assuming that words share the same functions to compose phrases. Thus, instead of learning an individual embedding for each n-gram in the corpus, the model only needs to learn the CNN weights for combining word-level embeddings, which have much fewer parameters.

Cross-match Layer: `Conv-KNRM` then matches query n-grams and document n-grams of different lengths in the embedding space. For query n-grams of length h_q and document n-grams of length h_d, a translation matrix M^{h_q,h_d} is constructed. Its elements are the similarity scores between the corresponding query-document n-gram pairs.

$$M_{i,j}^{h_q,h_d} = \cos\left(\vec{g}_i^{h_q}, \vec{g}_j^{h_d}\right) \tag{4.5}$$

The unified embedding representations allow cross-matching n-grams of different lengths, e.g., the query trigram"convolutional neural networks" and the document bigram "deep learning". It generates h_{max}^2 translation matrices.

$$\mathcal{M} = \left\{ M^{h_q,h_d} | 1 \leq h_q \leq h_{max}, 1 \leq h_d \leq h_{max} \right\} \tag{4.6}$$

4.1.2 Ranking with N-Gram Translations

`Conv-KNRM` then employs the `K-NRM` architecture (Chapter 3) to generate n-gram soft-match features. Kernel-pooling (Section 3.1) is applied to each M^{h_q,h_d} matrix in $\mathcal{M}$ to generate the soft-TF feature vector $\phi\left(M^{h_q,h_d}\right)$, which describes the distribution of match scores between query h_q-grams and document h_d-grams. This leads to the ranking features as follows.

$$\Phi\left(\mathcal{M}\right) = \phi\left(M^{1,1}\right) \oplus ... \oplus \phi\left(M^{h_q,h_d}\right) \oplus ... \oplus \phi\left(M^{h_{max},h_{max}}\right)$$

$\Phi(\mathcal{M})$ has $K \times h_{max}^2$ dimensions, K soft-TF features for each of the h_{max}^2 translation matrices in $\mathcal{M}$.

The learning-to-rank (LeToR) layer combines the soft-TF ranking features $\Phi(\mathcal{M})$ into a ranking score: $f(q,d) = \tanh\left(w_r^T \Phi(\mathcal{M}) + b_r\right)$, where w_r and b_r learns a linear combination of the n-gram soft-TF features. Same with `K-NRM`, standard pairwise learning-to-rank loss function is used to train the model (Section 3.1).

All of `Conv-KNRM` layers are differentiable; the whole model, including word embeddings $(\mathcal{V})$, CNN filters (W_h, b_h), and learning-to-rank layers (w_r, b_r) can be learned end-to-end from training data. For a model with vocabulary size $|V|$, L-dimensional word embeddings, F filters, h_{max} maximum n-gram length and K kernels, the embedding layer has $|V| \times L$ parameters, the CNN layer has $O(|h_{max}|LF)$ parameters, and the learning-to-rank layer has $K \times h_{max} + 1$ parameters.

The main capacity of the model is in the word embedding and CNN filters. They are expected to learn the word embeddings and n-gram compositions from training data and provide desired multi-level n-gram soft matches. The learning-to-rank layer serves as a linear feature combiner as in standard feature-based ranking.

4.1.3 Summary of Conv-KNRM Architecture

`Conv-KNRM` adds the ability of soft matching n-grams to `K-NRM` (Chapter 3) using convolutional neural networks (CNNs). Without CNNs, `Conv-KNRM` falls back to `K-NRM`.

Matching n-grams is a well-established idea in information retrieval. However, n-grams are usually treated identically to words: they are atomic index terms, have distinct term frequencies, use the same weighting function as unigrams, and must match exactly in query and document (Croft et al., 2009). If that approach is used by a neural ranker, the number of parameters to be learned can grow very large. The neural model has to deal with data sparsity and low efficiency problems, which are even harder for longer n-grams. `Conv-KNRM` avoids these problems by using CNNs to compose n-grams without dramatically enlarging the parameter space. It makes soft-matching n-grams convenient and efficient.

Table 4.1: characteristics of Sogou-Log and Bing-Log. Sogou-Log is the same as Table 3.1

Dataset	Sogou-Log		Bing-Log	
	Training	**Testing**	**Training**	**Testing**
Language	Chinese		English	
Document Fields	Title		Title, Snippet	
Queries	95,229	1,000	99,043	1,000
Documents Per Query	12.17	30.50	50	50
Search Sessions	31M	4M	31M	4M
Vocabulary Size	165,877	19,079	131,225	41,940

4.2 Domain Adaptation

End-to-end training `Conv-KNRM` requires large-scale training data, for example, user clicks in a commercial search log or industry-scale annotations (Mitra et al., 2017). Such data maybe available to a few commercial search engines like Google, Bing, or Sogou. But for many search domains, such data are not available. Collecting such labels also take time, so that new search verticals will face cold-start issues. To bring `Conv-KNRM` into wider adoption, we propose a domain adaption strategy. It learns `Conv-KNRM` from a source domain that has sufficient training data, and then re-trains its learning-to-rank layer in the target domain with limited labels.

The parameters of the embedding and convolution layers are learned in the source domain to absorb the rich relevance signals in the training data. They are then used in the target domain to generate soft-TF features. The kernels are capturing different types of soft match. For example, one kernel may count near-synonyms (e.g., "dog" and "puppy"); another kernel may count word pairs from the same concept class (e.g., "dog" and "cat", "Java" and "C++"). These soft match patterns are likely to be stable across related domains.

The learning-to-rank parameters indicate the importance of each kernel. They are re-trained on the target domain, because the importance of each type of soft matches can change over domains. For instance, our analysis in Chapter 3 shows that the exact match kernel is of low importance in search logs because the initial ranking produced candidate documents that already contain the query words; however, exact match can be a strong signal with a weaker initial ranker or in a recall-oriented domain.

Re-training the ranking layer in the target domain is a standard feature based learning-to-rank tasks. This allows one to add domain-specific features from the target domain. There is also no limitation on which learning to rank model to use in the target domain. One can leverage the power of any learning-to-rank model such as RankSVM (Joachims, 2002), ListMLE (Xia et al., 2008), and LambdaMART (Wu et al., 2010).

4.3 Experimental Methodology

This section describes our datasets, how training and testing were performed, our baseline algorithms, and implementation details.

4.3.1 Datasets

`Conv-KNRM` was evaluated using two search logs in different languages (Sogou-Log, Bing-Log), and an academic benchmark dataset from TREC (ClueWeb09-B).

Table 4.2: Training and testing labels for each dataset. DCTR used the DCTR click model to infer scores that were mapped to 5 Likert scales (Chuklin et al., 2015). Clicks used the sole click in a session as the binary label. TREC labels were the 5 official grades.

Dataset	Train	Test
Sogou-Log	DCTR	Testing-SAME: DCTR Testing-RAW: Clicks
Bing-Log	DCTR	Testing-SAME: DCTR Testing-RAW: Clicks
ClueWeb09-B	Embedding & CNN: Bing-Log, DCTR	TREC labels
	LeToR: TREC labels	

Sogou-Log is the same dataset used in our previous work on K-NRM (Table 3.1). Bing-Log is a one-month sample of a 2006 Bing log from the WSDM 2009 Web Search Click Data Workshop. Table 4.1 compares the characteristics of Sogou-Log and Bing-Log. Bing-Log contains the top 50 URLs for each query, and clicked URLs in each session. Following Sogou-Log, we split the Bing sessions into training and testing sets with no overlapping queries. Test queries were sampled uniformly because the log contained few head queries. Bing-Log includes documents' titles and snippets. Most snippets had 30-50 words. We can not crawl enough body texts because URLs were from 2006. All texts were tokenized and lower-cased.

ClueWeb09-B is used for domain adaptation experiments. The ClueWeb09-B corpus contains about 50 million English web documents from 2009. The TREC 2009-2012 Web Tracks created 200 queries and corresponding relevance judgments. We followed a standard re-ranking methodology in prior research (Dalton et al., 2014; Xiong et al., 2017a): re-rank the top 100 candidate documents retrieved by Galago using sequential dependency model queries; the INQUERY stopword list augmented with web-specific stop words; KStemming; and spam filtering using Waterloo spam score (Cormack et al., 2011) with threshold 60. Documents were parsed by Boilerpipe (Kohlschütter et al., 2010) using the "KeepEverytingExtractor". The title and the first 50 words in the body field were used to be more consistent with the title and snippet setting of the source domain (Bing-Log).

4.3.2 In-Domain Training and Testing

Training and testing labels on Sogou-Log and Bing-Log were generated with the same setting as in Chapter 3 using clicks. The training labels were generated by the DCTR click model (Chuklin et al., 2015) from user clicks in the training sessions, and training preference pairs were constructed accordingly. Two of the three testing conditions described in Chapter 3 were used in this work. Testing-SAME generates testing relevance labels with DCTR, same as for generating training data. This setting evaluates the model's ability to fit explicit user preferences. Testing-RAW only considers the clicked document in a single-click session as relevant. 57% of Sogou testing sessions had only one click. 92% of Bing testing sessions had only one click. The Testing-DIFF condition was omitted as it produced results similar to Testing-RAW.

4.3.3 Domain Adaptation Training and Testing

The National Institute of Standards and Technology (NIST) provides 200 queries and corresponding relevance judgments for ClueWeb09-B. The domain adaptation experiment tests how well the n-gram soft matches trained from one domain (Bing-Log) generalize to a similar domain (ClueWeb09-B). Both

datasets contain English web documents, the timespans are somewhat similar (2006 vs. 2009), and TREC queries are similar to Bing queries[2]. However, the two datasets have different documents, different indexing methods, and different the initial rankers (Bing vs. Galago); the relevance labels are also rather different (clicks vs. manual assessments).

On ClueWeb09-B, Conv-KNRM was first trained on the Bing-Log (as described above). Then the embeddings and convolution filters were "frozen" and soft TF-features $\Phi(\mathcal{M})$ were extracted using the same kernels for ClueWeb09-B. We also included the initial retrieval score from Galago with sequential dependency model (Galago-SDM) to provide whole-document information as Conv-KNRM only uses the title and the first 50 words of the body. The learning-to-rank parameters were retrained and tested using TREC relevance judgments (Table 4.2), 10-fold cross-validation, and RankSVM to add regularization.

4.3.4 Baselines

The first group of baselines are traditional IR baselines. For Sogou-Log and Bing-Log, these are the same as Chapter 3. They includes standard unsupervised retrieval models (BM25 and Lm) and feature-based learning-to-rank models (RankSVM and Coor-Ascent). Retrieval models or feature extractions were applied on the full text of Sogou documents, or the title plus snippet of Bing documents.

For ClueWeb09-B, we used state-of-the-art baselines from prior research (Xiong et al., 2017a) [3]. The baselines include Indri's language model (Indri), Galago with sequential dependency model queries (Galago-SDM), and learning-to-rank models: RankSVM and Coor-Ascent.

The second group of baselines are Neural IR baselines. They included two strong baselines from Chapter 3: CDSSM (Shen et al., 2014b) and DRMM (Guo et al., 2016a). In addition, MatchPyramid (Pang et al., 2016b) was included. MatchPyramid (Pang et al., 2016b) is an interaction based models built upon the embedding translation matrix. MatchPyramid uses CNNs to combine the translation scores to the ranking score. In comparison, the proposed Conv-KNRM use CNNs on the word embeddings. Finally, K-NRM (Chapter 3) was also used as a baseline.

Among these neural IR baselines, DRMM and K-NRM were compared on the ClueWeb09-B dataset. DRMM uses fixed embeddings and only learns the learning-to-rank layers, and can be trained with limited training data. K-NRM was tested the same as Conv-KNRM in the domain adaption fashion. CDSSM and MatchPyramid performed worse than DRMM on TREC data in previous studies (Pang et al., 2016a; Guo et al., 2016a).

4.3.5 Implementation Details

All supervised traditional IR models were trained and tested using cross-validation on the testing data. On search logs, 5-fold cross validation were used to be consistent with the previous study in Chapter 3. On ClueWeb09-B, the 10-fold cross validation splits from the provided baselines were used. All RankSVM's used the linear kernel with the hyper-parameter C selected from the range $[0.0001, 10]$ on the development set. Recommended settings of Coor-Ascent were kept. All neural IR methods are trained on the training splits. On ClueWeb09-B, DRMM was cross-validated; K-NRM and Conv-KNRM was pretrained on Bing-Log, then used RankSVM with cross-validation to retrain the learning-to-rank layer.

Some experiments used multiple document fields as input. On Sogou-Log, traditional IR methods used both title and body, and neural IR methods only used title, as discussed in section 5.1. On Bing-Log, all methods used the title and snippets. On ClueWeb09-B, all methods used title and body, except K-NRM

[2]NIST sampled the TREC queries from a Bing search log. We removed TREC queries from our Bing training data. Our training and testing data have no queries in common.

[3]https://boston.lti.cs.cmu.edu/appendices/SIGIR2017_word_entity_duet/

and Conv-KNRM which used title and first 50 words in the body as snippets, to be consistent with the source domain. When multiple fields were used, a separate set of features from each field was generated; the combination weights were learned as well.

All neural IR models used word embeddings. DRMM used pre-trained word2vec embeddings from the candidate documents in the search log, or the ClueWeb corpus. MatchPyramid, K-NRM, and Conv-KNRM embeddings were all learned end-to-end using the query logs. For Sogou-log, we followed the settings in Chapter 3 and set embedding dimension $L = 300$. For Bing-Log, we set $L = 100$ because our pilot study showed that $L = 100$ has similar performance with $L = 300$ but the training is 3 times faster.

The hyperparamters of Conv-KNRM were configured as the follows. N-gram lengths were $h = 1, 2, 3$. Longer n-grams with $h > 3$ usually exceed the length of web search queries. The number of CNN filters F was 128; we found that F in the range of $(50, 300)$ give similar results. The other hyper parameters and implementation details all followed the settings used in Chapter 3. We released the trained models for reproducibility [4].

4.4 Evaluation Results

Three experiments were conducted to analyze Conv-KNRM's performance: its ranking accuracy when trained end-to-end, contributions of n-gram soft match, and the effectiveness when adapted to new domain.

4.4.1 End-to-End Accuracy

Table 4.3 reports the ranking accuracy of Conv-KNRM and the baselines on the Sogou-Log and Bing-Log.

On Testing-SAME, Conv-KNRM outperformed all baselines by large margin with statistical significance. The closest baseline is K-NRM, the non-convolutional version of Conv-KNRM, but the differences were still large. Conv-KNRM performed better in higher ranking positions: its NDCG@1 almost doubled Coor-Ascent, a strong feature based learning-to-rank system. These results show Conv-KNRM's effectiveness when trained and tested on the same labeling scenario.

Testing-Raw evaluates the model's performance by raw user clicks. The same stable improvements of Conv-KNRM over all baselines were observed. Since this evaluation uses sessions with only one click, the MRR scores directly reflect the reciprocal rank of user-clicked documents. On Sogou-Log, the average rank of clicked documents of all methods except K-NRM and Conv-KNRM was below rank 5. K-NRM pulled the clicked document to rank 3, and Conv-KNRM further promoted it to rank 2.7. On Bing-Log, Conv-KNRM pulled the clicked document of all methods more than 1 position higher.

The only neural IR baselines that outperformed feature-based learning-to-rank are the two *interaction* based and *end-to-end* trained ones: MatchPyramid and K-NRM. Although other neural IR methods can improve over unsupervised baselines, feature-based learning-to-rank methods are harder to beat; end-to-end learned embeddings and match-based techniques are necessary for current neural IR methods to provide additional improvements (Pang et al., 2016a; Zamani and Croft, 2017)

Comparing the two strong neural IR baselines, K-NRM outperforms MatchPyramid by a large margin. Both methods use end-to-end learned word embeddings to build the translation matrix. The difference is that K-NRM uses kernel-pooling to summarize 'soft-TF' counts from the translation matrix, while MatchPyramid directly applies the CNN to combine the translation scores. Our results demonstrate that counting soft matches are more effective than weight-summing the similarities, showing the advan-

[4] http://boston.lti.cs.cmu.edu/appendices/WSDM2018-ConvKNRM/

Table 4.3: Ranking accuracy of Conv-KNRM and baseline methods. Relative performances compared with K-NRM are in percentages. $\dagger, \ddagger, \S, \P, *$ indicate statistically significant improvements over Coor-Ast†, DRMM‡, CDSSM§, MatchPyramid¶ and K-NRM*, respectively.

(a) Sogou-Log

Method	Testing-SAME				Testing-Raw	
	NDCG@1		NDCG@10		MRR	
BM25	0.142	-45%	0.287	-34%	0.228	-33%
RankSVM	0.146	-44%	0.309	-29%	0.224	-34%
Coor-Ascent	$0.169^{\ddagger\S}$	-34%	$0.355^{\ddagger\S}$	-16%	0.242	-29%
DRMM	0.137	-51%	0.315	-27%	0.234	-31%
CDSSM	0.144	-44%	$0.333^{\ddagger}$	-23%	0.232	-32%
MatchPyramid	$0.218^{\dagger\ddagger\S}$	-15%	$0.379^{\dagger\ddagger\S}$	-12%	0.240	-29%
K-NRM	$0.264^{\dagger\ddagger\S\P}$	$--$	$0.428^{\dagger\ddagger\S\P}$	$--$	$0.338^{\dagger\ddagger\S\P}$	$--$
Conv-KNRM	$\mathbf{0.336}^{\dagger\ddagger\S\P*}$	$+30\%$	$\mathbf{0.481}^{\dagger\ddagger\S\P*}$	$+11\%$	$\mathbf{0.358}^{\dagger\ddagger\S\P*}$	$+5\%$

(b) Bing-Log

Method	Testing-SAME				Testing-Raw	
	NDCG@1		NDCG@10		MRR	
BM25	0.043	-79%	0.123	-63%	0.102	-61%
RankSVM	0.128	-39%	$0.266^{\ddagger}$	-20%	0.207	-22%
Coor-Ascent	0.142	-32%	$0.268^{\ddagger}$	-20%	0.208	-22%
DRMM	0.137	-34%	0.247	-26%	0.200	-25%
CDSSM	0.156	-25%	0.273	-18%	0.212	-20%
MatchPyramid	$0.182^{\dagger\ddagger\S}$	-12%	$0.301^{\dagger\ddagger\S}$	-10%	$0.244^{\dagger\ddagger\S}$	-8%
K-NRM	$0.208^{\dagger\ddagger\S\P}$	$--$	$0.334^{\dagger\ddagger\S\P}$	$--$	$0.265^{\dagger\ddagger\S\P}$	$--$
Conv-KNRM	$\mathbf{0.300}^{\dagger\ddagger\S\P*}$	$+44\%$	$\mathbf{0.437}^{\dagger\ddagger\S\P*}$	$+31\%$	$\mathbf{0.354}^{\dagger\ddagger\S\P*}$	$+34\%$

Table 4.4: Ranking accuracy of Conv-KNRM variants. Relative performances compared with Unigram-only model (K-NRM) are in percentages. †, ‡, §, ¶ indicate statistically significant improvements over Unigram†, +Bigram‡, +Trigram§ and +Uni-x-Bi¶, respectively.

(a) Sogou-Log

Conv-KNRM Variant	**Testing-SAME**				**Testing-Raw**	
	NDCG@1		**NDCG@10**		**MRR**	
Unigram	0.264	−−	0.428	−−	0.338	−−
+Bigram	$0.287^{\dagger}$	+11%	0.442	+2%	0.314	−8%
+Trigram	$0.286^{\dagger}$	+11%	$0.454^{\dagger}$	+5%	$0.330^{\ddagger}$	−3%
+Uni-x-Bi	$0.308^{\dagger}$	+19%	$0.458^{\dagger\ddagger}$	+6%	$0.346^{\ddagger\S}$	+2%
Full Model	$0.336^{\dagger\ddagger\S\P}$	+30%	$0.481^{\dagger\ddagger\S\P}$	+11%	$0.358^{\dagger\ddagger\S}$	+5%

(b) Bing-Log

Conv-KNRM Variant	**Testing-SAME**				**Testing-Raw**	
	NDCG@1		**NDCG@10**		**MRR**	
Unigram	0.208	−−	0.334	−−	0.265	−−
+Bigram	$0.235^{\dagger}$	+13%	$0.385^{\dagger}$	+15%	$0.301^{\dagger}$	+14%
+Trigram	$0.252^{\dagger}$	+21%	$0.399^{\dagger}$	+20%	$0.318^{\dagger\ddagger}$	+20%
+Uni-x-Bi	$0.275^{\dagger\ddagger}$	+32%	$0.417^{\dagger\ddagger\S}$	+25%	$0.335^{\dagger\ddagger\S}$	+26%
Full Model	$\mathbf{0.300}^{\dagger\ddagger\S\P}$	+44%	$\mathbf{0.437}^{\dagger\ddagger\S\P}$	+31%	$\mathbf{0.354}^{\dagger\ddagger\S\P}$	+34%

tages of kernel-pooling over other commonly-used neural network layers, such as CNNs and feed-forward layers.

Recall that Conv-KNRM is a richer model than K-NRM only because it leverages convolutional neural networks to learn the n-gram compositions and thus enable n-gram soft matches. The improvements of Conv-KNRM over K-NRM reveal the advantage of n-gram soft matches. The relative improvements on Sogou and Bing also correlate with our intuitions of n-gram's importance in Chinese and English. In Chinese, words are segmented by word segmentation tools. An important goal of Chinese word segmentation research is to cut meaningful phrases into one word. For example, "information retrieval", "deep learning", and "The People's Republic of China" are all unigrams in Chinese. As a result, the gains are much larger on English than on Chinese.

4.4.2 Contribution of N-Gram Soft-match

This experiment studied the contribution of n-gram soft matches by comparing several Conv-KNRM's variations. Conv-KNRM composes n-grams with lengths up to h_{max} and cross-matches them in a unified embedding space. We started with K-NRM, which is Conv-KNRM without CNNs, and incrementally added bigram matches (+Bigram), trigram matches (+Trigram), cross unigram-bigram matches (+Uni-x-Bi), and cross all three n-grams' matches which is the Full Model. Results are shown in Table 4.4.

Longer n-gram were more effective in English. On the Bing-Log, trigrams were better than bigrams, and bigrams were better than unigrams. The effect was weaker in Chinese, with mixed performances on different settings. In Chinese, words are segmented by word segmentation tools. An important goal of Chinese word segmentation research is to cut meaningful phrases into one word. For example, 'informa-

Table 4.5: Performance of conv-KNRM on ClueWeb09-B using domain adaptation. Relative performance in percentages are compared to Coor-Ascent. W(in)/T(ie)/L(oss) to Coor-Ascent are compared at NDCG@20. $\dagger, \ddagger, \S, \P*$ indicate statistically significant improvements over Indri†, Galago SDM‡, RankSVM§, Coor-Ascent ¶ and DRMM+SDM *.

Method	ClueWeb09-B						
	NDCG@1		NDCG@10		NDCG@20		W/T/L
Indri	0.239	−6%	0.229	−15%	0.236	−12%	68/31/101
Galago-SDM	0.219	−14%	0.238	−11%	0.250†	−7%	63/39/98
RankSVM	0.236	−7%	0.256†‡	−5%	0.263†‡*	−2%	82/47/71
Coor-Ascent	0.255†‡*	−−	0.268†‡	−−	0.268†‡	−−	–/–/–
DRMM+SDM	0.215	−16%	0.261†‡	−3%	0.243	−9%	66/34/100
K-NRM	0.235	−8%	0.264†‡	−2%	0.269†‡*	+0%	69/42/89
Conv-KNRM-exact	0.231	−10%	0.263†‡	−2%	0.270†‡*	+1%	78/42/80
Conv-KNRM	**0.294**†‡§¶*	+15%	**0.289**†‡§¶*	+8%	0.287†‡§¶*	+7%	88/38/74

tion retrieval', 'deep learning', and 'thee People's Republic of China' are all unigrams in Chinese. As a result, the gains are much larger on English than on Chinese.

Cross matching n-grams of different lengths boosted accuracy in both languages. +Uni-x-Bi performed significantly better than +Bigram on most metrics, and Full Model outperformed all other variants significantly. Cross matching is effective because related concepts do not necessarily have the same length, e.g. "FIFA" and "world cup". Composing n-grams using CNNs makes cross matching simple: all n-grams, despite with different lengths, are represented and matched in the same embedding space.

4.4.3 Domain Adaption

Our third experiment examined the effectiveness of Conv-KNRM when adapted to a domain where large scale training data is not available. We trained Conv-KNRM's word embeddings and convolution filters in Bing-Log with a large amount of user preference labels from clicks, and re-trained the learning-to-rank part on ClueWeb09-B's TREC ranking labels. Table 4.5 reports the results.

Indri and Galago-SDM are unsupervised baselines. RankSVM, Coor-Ascent, and DRMM+SDM are supervised baselines. They used in-domain training with cross-validation, because they are relatively small models that can be learned with TREC-scale training labels. DRMM+SDM is a variant of DRMM that uses Galago-SDM score as an additional feature; it showed higher performance than the standard DRMMM. Conv-KNRM and K-NRM were both trained using domain adaption. We also examined Conv-KNRM-exact which only uses exact-matches of n-grams, e.g. "world cup" can only be matched to "world cup".

As shown in Table 4.5, K-NRM was not able to beat DRMM+SDM, meaning that the effectiveness of unigram level soft matches was weakened by domain differences. Conv-KNRM-exact performed about the same, and was weaker than learning-to-rank approaches. It does no more than exact phrase matching as in SDM. Conv-KNRM differs from K-NRM and Conv-KNRM-exact by soft-matching n-grams; it outperformed the two strong traditional learning-to-rank models. The results demonstrate that the learned n-gram soft matches of Conv-KNRM can generate to a different domain.

Feature Weight Analysis: To further study generalization ability of Conv-KNRM, we investigated the importance of Conv-KNRM's soft-TF features ($\Phi(\mathcal{M})$) in the adapted model. If the soft n-gram

Table 4.6: Examples of matched n-grams between query and snippets. Black phrases contribute more to the relevance score than gray ones.

Query	Snippet
sewing instructions	...newsletter sewing ideas...quilting 101 what is a quilt...
atypical squamous cells	...treatment decision tools cervical cancer : prevention...
moths	...commonly known as the 'smaller moths ' (micro , lepidoptera)...
fickle creek farm	...extended stay lodging rv parks where to eat & drink nightlife ...
university phoenix	degree masters degrees account degrees business degree...
wedding budget calculator	...photographs bridal board my perfect planner tools my check lists...

matching patterns learned from the Bing-Log is generalizable, their soft-TF feature weights would share a reasonable chunk of learning to rank weights in the adapted model.

We verified this by analyzing the weights RankSVM assigned to different groups of ranking features. Results are shown in Figure 4.2. Each analysis divides features into two groups, e.g. exact match features and soft match features. It then calculates the percentage of weight given to each type of features by summing up the absolute weight values of the feature set. In Figure 4.2, most of the weight goes to soft matches (Exact v.s. Soft); N-gram matches have more weight than unigram matches (Unigram vs. N-gram); and matching n-grams of different lengths is important compared to matching n-grams with same length (Same-length v.s. Cross-length). The high feature weights on soft n-gram match features reveals that these features do provide useful information to the learning-to-rank model—more useful than n-gram exact matches, despite that the n-gram soft matches were trained and tested on two rather different domains: different labels, different documents, and non-overlapping queries.

Case Studies: We performed case studies to better understand the soft n-gram matching. Table 4.6 shows examples of relevant documents that are correctly placed at rank 1 by `Conv-KNRM`, but not by `RankSVM` and `KNRM`. By sorting n-gram pairs according to each kernel feature's individual performance, we can find the most important soft match that makes the document ranked highly, as highlighted in Table 4.6. These cases demonstrated the effectiveness in `Conv-KNRM`. First, `Conv-KNRM` overcomes the lexical mismatch, and finds query-document connections that are difficult for exact-match-based approaches, e.g., "sewing instructions" and "quilting 101". Second, `Conv-KNRM` captures n-gram matches that are different with word matches like `K-NRM`. For example, ("atypical squamous", "cervical cancer") is a strong match, but the connection between their unigram pairs, e.g. ("atypical", "cervical"), are much weaker. These examples also illustrates `Conv-KNRM`'s generalizability: the matchings make sense in various contexts.

In summary, the domain adaptation experiment provides a thorough view of the generalization ability of `Conv-KNRM`. The evaluation on ClueWeb09-B shows that the cross-domain soft n-gram matching provides significant gains over in-domain learning-to-rank. Feature weight analysis demonstrates that the adapted model puts the majority of feature weights on n-gram soft match signals, meaning that these soft match patterns do provide useful ranking evidence in the new domain. Case studies prove that the learned soft-matches are intuitive and cover universally meaningful information needs. To the best of our knowledge, this is the first time we have seen such generalization ability in neural IR models.

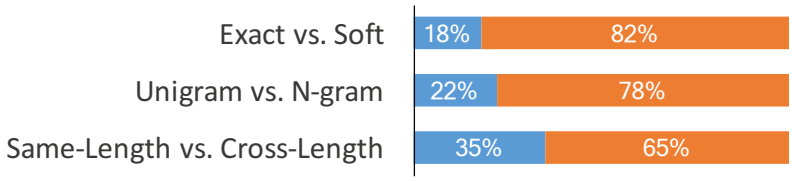


Figure 4.2: Learned weights of different parts of ranking features on ClueWeb09-B. The percentage is the fraction of absolute weights on each side learned by linear RankSVM.

4.5 Conv-KNRM Summary

This chapter presents Conv-KNRM, a convolutional kernel-based neural ranking model that models n-gram soft matches for information retrieval. It is based on the success of K-NRM (Chapter 3), and adds n-grams soft match signals into the framework using Convolutional Neural Networks. Different from typical approaches that treats n-grams as discrete index terms, Conv-KNRM uses Convolutional Neural Networks to compose n-gram embeddings from word embeddings, addressing the problems of dimension explosion and data sparsity. Conv-KNRM then cross-matches n-grams of various lengths in the unified embedding space, applies kernel pooling to group soft match signals, and uses learning-to-rank to obtain the final ranking score.

Experiments on Chinese and English search logs demonstrate the advantages of soft-matching n-grams in relevance ranking. Conv-GRAM almost doubled the NDCG@1 scores compared to feature-based ranking approaches, and outperformed the previous state-of-the-art model by over 30% at the top. Trained end-to-end with user feedback, Conv-KNRM learns n-gram soft match patterns tailored for matching queries and relevant documents, for example, the query "farm" is matched to "eat & drink". Such IR-customized n-gram soft-match has not been seen much in previous work.

Based on the analysis, the key to Conv-KNRM's advantages is cross-matching n-grams of different lengths. Cross-matching consistently outperformed its non-cross-matching variants. On the Chinese search log, Conv-KNRM without cross-matching is about the same as its unigram competitor K-NRM, due to the phrase-like characteristics of Chinese unigrams. Cross-matching is important because related concepts do not necessarily have the same number of words, for instance, "deep learning" and "convolutional neural network". But there has been little study on it due to the limitation of discrete n-gram representation. The CNN approach of modeling n-grams makes cross-matching feasible, efficient, and effective.

Beyond the good performance when trained end-to-end in domain, this chapter also develops methods to improve the generalization ability of Conv-KNRM. The model learned from Bing-log significantly outperformed strong learning-to-rank baselines when adapted to TREC Web Track task, despite important domain differences including corpus, queries and evaluation conditions. Experiments show that the embedding and CNN layers can be directly used in another related domain to generate n-gram soft-matching features. Further analysis explains the generalizability: the learned n-gram soft-matching patterns encode universal properties of language usage in ad hoc search tasks, and provide important evidences for relevance ranking even when used across domains.

Our feature weight analysis demonstrates the importance of soft match features: over 80% weights were assigned to soft match, while only 20% were assigned to exact match; 65% weights were assigned to cross-length match, while only 35% were to same-length match. The soft match signals that take up the majority of importance were not available in classic information retrieval. K-NRM and Conv-KNRM unlocked these features, successfully broke the long-standing "exact match" bottleneck in IR.

Chapter 5

Adapting Conv-KNRM from Text to Engineering Diagrams[1]

People search a wide range of data every day – articles, images, products, people's social-media accounts, etc. Traditionally, each type of data require its own data representation, feature extractor, and corresponding retrieval system. Although those retrieval systems may face many common issues, e.g., how to model soft match, the distinct data representations and features makes it prohibitive to re-use a retrieval system on a different media.

Neural networks make it possible to use similar architectures for different tasks. Under distributed representations, queries and documents, no matter they are natural languages or images, can be converted into a unified representation, enabling the model to deal with a universal representation without worrying about the original data type. With end-to-end learning, features can be learned automatically, greatly reducing the efforts on domain-specific feature engineering. To maximize the potential of adapting across search tasks, it is of great interest to develop neural architectures that address fundamental problems in search, so that one can re-use them on different tasks instead of designing new models every single time.

Chapter 3 and 4 show the ability of `K-NRM` and `Conv-KNRM` to search text documents. This chapter challenges the generalization ability of these neural ranking architectures on a very different task – searching engineering diagram images.

Engineering diagram search is a practical task. In large engineering organizations that manufacture complex products, many engineers at different locations may collaborate to design a product. Searching a corpus of engineering diagrams for similar parts helps staff estimate production costs for new parts based on the costs of similar existing parts; and reduce production costs by using existing parts or replacing several slightly different parts with a new, more widely applicable part. Engineering diagrams use line drawings and metadata such as part reference numbers to represent the elements of a part or an assembly of parts. Often they are created by computer-aided design (CAD) systems. However, the detailed descriptions stored in CAD systems are unwieldy, often unnecessary, and sometimes too sensitive to be shared widely within an organization. Thus, engineering diagrams – *images* – are the main method of describing parts and assemblies for many tasks. Finding engineering diagrams that contain a specific part or assembly is a type of *image retrieval*.

Image retrieval is a long-studied research area, but engineering diagrams have its unique characteristics. Most prior research focuses on pictures, and specifically how to represent pictures so that similar

[1]This chapter is based in full on a previously published paper (Dai et al., 2019) appearing in the Web Conference 2019. This work was led by Dai and Callan. The Ikea dataset 5.4 used in this paper was built by Fan. The implementation, experiments, and analysis were jointly shared by Dai, Fan, and Rahman.

pictures are easy to recognize. The majority of previous work uses *representation-based* models, which use a single representation of the entire image, and use a simple scoring function (e.g., cosine) to measure the relevance between them. *Representation-based* approaches have many advantages: they are fast, suppress or generalize unnecessary detail, and describe the main elements of an image. However, in engineering diagram search, the query diagram is often expected to match only a small region of an engineering diagram, for example, to find every diagram that uses a specific part. Thus, it is more of a local matching problem that requires capturing *local interactions* between small parts. Moreover, typical engineering diagrams have a visual syntax and contain visual metadata elements, for example, locator elements that place the diagram within a larger context; callouts that provide other perspectives; bubbles that provide additional detail; and lines that illustrate relationships among selected parts. It is an open question whether existing representation-based image retrieval algorithms are effective for engineering diagram search.

The above characteristics make engineering diagrams search similar to text search problems – people use a small part (a few words) to retrieve documents that mentions these parts (words) or their closely-related parts (soft match). Some visual elements, such as locators and callouts, resembles the stopwords and off-topic words in documents. Neural ranking models like `K-NRM` (Chapter 3) and `Conv-KRNM` (Chapter 4) were designed for modeling local query-document interactions, and might be a potential solution to engineering diagram search. This chapter adapts these neural ranking architectures that are designed for text to the engineering diagram search problem.

One of our approaches is an unsupervised ranker called **DI**agram **S**earc**H** with Local Histogram Pooling Matching Network (`DISH-HP`). It is derived from a prior neural ranking model, `DRMM` (Guo et al., 2016a) – the system that inspired `K-RNM`. `DISH-HP` models the relevance of a diagram to a query image based on the similarity of local regions, and uses histogram pooling to combine evidence (Guo et al., 2016a). It first extracts local convolution features from the middle layers of a pretrained network, such as VGG-16 (Visual Geometry Group, Department of Engineering Science, University of Oxford, 2018). Each convolution feature represents a small region in the original diagram, which are treated as "words". `DISH-HP` calculates the similarity scores between every pair of query and document regions. It then groups the region pairs into bins according to their similarity scores.

The second of our new approaches is called Diagram Search with Local Convolutional Matching Network (`DISH-Conv`). It was derived from `Conv-KNRM` (Chapter 4). `Conv-KNRM` use 1-D convolutional neural networks to build n-gram embeddings from word embeddings. Similarly, the `DISH-Conv` model employs 2-D convolutional neural networks to build feature vectors for larger regions from small regions, generating multi-scale region representations. A cross-scale matching layer calculates the similarity between regions of different scales, enabling a smaller region in the query to match to a larger region in the candidate diagram. This is similar to the cross-matching layer used in `Conv-KNRM` that matches n-grams of different lengths. `DISH-Conv` can be trained end-to-end with relevance signals, so that the convolution, matching, and gating are optimized towards the specific dataset and task.

`DISH-HP` and `DISH-Conv` can be used in a pipelined re-ranking architecture. `DISH-HP` is the first stage that quickly ranks images. The top N candidates are passed to `DISH-Conv` to refine the ranking. N is limited to a few hundred or a few thousand, so that the second stage is able to use a more accurate ranking model that would be too computationally expensive for the first stage brute-force search.

Evaluation was done on two datasets. The first dataset is a new diagram search dataset we developed by crawling furniture assembly manuals from the Ikea website. Experiments on large-scale auto-generated relevance labels show that `DISH-HP` is more accurate than several unsupervised rankers that use global image representations. Using `DISH-Conv` to re-rank `DISH-HP` results improves accuracy substantially. Larger improvements over baseline models are shown on more difficult queries that have rotation and

scale changes. Evaluation with a small set of manual relevance assessments further confirms these results. The second dataset was provided by the Boeing company, which consists of engineering diagrams of the Boeing 777 interior designs. Evaluation on the Boeing dataset confirmed the results, and demonstrated that the proposed models are robust across two distinct domains.

The rest of this chapter is organized as follows. Sections 5.1 gives a brief overview of related work. Section 5.2 and 5.3 introduce the how we adapt DRMM and Conv-KNRN into DISH-HP and DISH-Conv. Section 5.4 discusses the Ikea diagram search dataset and our experimental methodology. Section 5.5 reports and discusses experimental results. Section 5.6 summarizes and concludes the chapter.

5.1 Related Work

The majority of work on image-based engineering diagram search was developed before deep neural networks was widely applied in image search. These systems often take several steps to first identify lines and shapes and then to calculate similarities between two diagrams. Task specific features and matching criteria were designed, leading to complex systems and extensive feature engineering (Müller and Rigoll, 1999; Fonseca et al., 2005). A more recent work by Eitz et al. (2011) demonstrates the effectiveness of bag-of-words models with SIFT-style features on line-drawings and sketch-based image retrieval. To the best of our knowledge, little exploration has been made on using deep neural networks for engineering diagram search.

Deep neural networks have been studied extensively in general-domain image search tasks. The vast majority of image search approaches are *representation-based*. The early CNN-based image retrieval work (Babenko et al., 2014; Gong et al., 2014) takes the activation of fully connected layers as *global* descriptors. Later work extracts convolution features from sliding windows at different image regions, and used max/sum pooling over the convolution features to produce the single image embedding (Babenko and Lempitsky, 2015; Tolias et al., 2015; Kalantidis et al., 2016; Jimenez et al., 2017). The relevance between two images is evaluated by the simple dot product or cosine similarity between the image embedding.

There exist a few *interaction-based* approaches to image search, mostly using patch-based methods. They represent an image using multiple image patches, with the goal to address the problem when the item of interest has appeared on a different scale at an arbitrary position in the relevant image (Gong et al., 2014; Razavian et al., 2014). However, patch-based methods are computationally expensive because each individual patch is resized to the same size as the original image and fed forward into a deep convolutional neural network to extract features.

Comparing to image search, interaction-based approaches have received more attention in text search. The DRMM model (Guo et al., 2016a) uses groups the word-word soft matches into a histogram based on the similarity score, and learns to combine the bins. It allows the model to take into consideration both strong word matches and weaker word matches. The K-NRM model(Chapter 3) integrates DRMM with the ability to learn customized embeddings. The Conv-KNRM model (Chapter 4) extends K-NRM with n-grams using convolutional neural networks. It uses different sizes of convolutions to build unigram, bigram and trigram embeddings from word embeddings. It is an open question if these interaction-based text ranking models can also be applied to images.

5.2 DRMM to DISH-HP: Local Histogram Pooling Matching Network

Interaction-based text retrieval models, which match a query to localized regions of a document, are more accurate than representation-based models because i) text queries are not expected to match all,

or even most, of a text document, ii) local interactions provide multiple and detailed match signals, and iii) irrelevant regions have no effect. Interaction-based models could be beneficial to diagram search by modeling the interactions of a local region from the query diagram and a local region from the document diagram. This section adapts DRMM (Guo et al., 2016a) – the model that inspired K-NRM (Chapter 3) – for engineering diagram search. The adapted model is DISH-HP, a *Local Histogram Pooling Matching Network* for engineering diagram search that models the relevance between two diagrams based on how similar their local regions ("words") are.

The input to DISH-HP is a list of local convolution feature vectors extracted from the query diagram Q and the document diagram D. In our experiments, features are extracted from the last convolution layer of the VGG-16 network (Visual Geometry Group, Department of Engineering Science, University of Oxford, 2018). This layer divides the image into $W \times H = 14 \times 14 = 196$ grids, each corresponding to a 16 pixel $\times$ 16 pixel rectangular region in the diagram. For each of the $H \times W$ regions, a $d = 512$ dimension feature vector is generated. Hence, Q and D are represented by matrices of size $[H \times W, d] = [196, 512]$.

Given the feature matrices of Q and D, DISH-HP collects the similarity scores between every pair of query region and document region. It then groups the region pairs into bins based on the similarity scores and builds a histogram. The histogram is pooled to produce a relevance score, where two diagrams are considered similar if many of the region pairs fall into the high similarity bins. More specifically, DISH-HP builds a similarity matrix M of size $[H \times W, H \times W]$ where the cell $M_{i,j}$ is the cosine similarity between the i-th query feature and the j-th document feature. A histogram is built for each row of the similarity matrix by assigning the similarity values into 11 bins:$B_0 = [1, 1]$, $B_1 = [0.8, 1]$, $B_2 = [0.6, 0.8)$, ..., $B_{10} = [-1, -0.8)]$. DISH-HP then counts the points in each bin. The counts indicate how many document regions are very similar to the query region (e.g., bin $[0.8, 1]$), how many are somewhat similar (e.g., bin $[0.6, 0.8)$), and how many are dissimilar. The number of points in the first K high-similarity bins $\{B_1, ..., B_K\}$ are taken as the score for the document diagram to match the i-th query region:

$$s(Q_i, D) = \sum_{j=1...K} |B_j|. \tag{5.1}$$

The final relevance score is the log-sum of counts from every query region:

$$s(Q, D) = \sum_{i=1...R} \log s(Q_i, D). \tag{5.2}$$

Log-sum squashes very high values into smaller values. It helps to avoid the matching being dominated by a few query regions that are repeatedly matched in the document diagram.

One issue with the above method is that the relevance score can be overwhelmed by noisy match signals from low-density regions. Engineering diagrams are often very sparse; blank or near-blank regions will dominate the high-similarity bins. To address this issue, features from low-density regions are filtered out before computing the similarity matrix. A feature vector is removed if its L2-norm is smaller than a threshold T, with the assumption that lower L2-norm indicates lower density of that region. The L2-norm threshold T is a hyper-parameter to be tuned.

In summary, DISH-HP explicitly models the similarity between local regions from the query and the document. DISH-HP is expected to retrieve more-relevant documents than global matching methods because the local, region-level matching provides more evidence than matching at the global image level. In terms of efficiency, DISH-HP is slower than global methods. DISH-HP needs to compute R^2 cosine similarity values for a document with R local regions; global methods only need to compute a single global cosine similarity. The bottleneck for both global and local method lies in extracting features from the query using a deep convolutions neural network, which cannot be done offline.

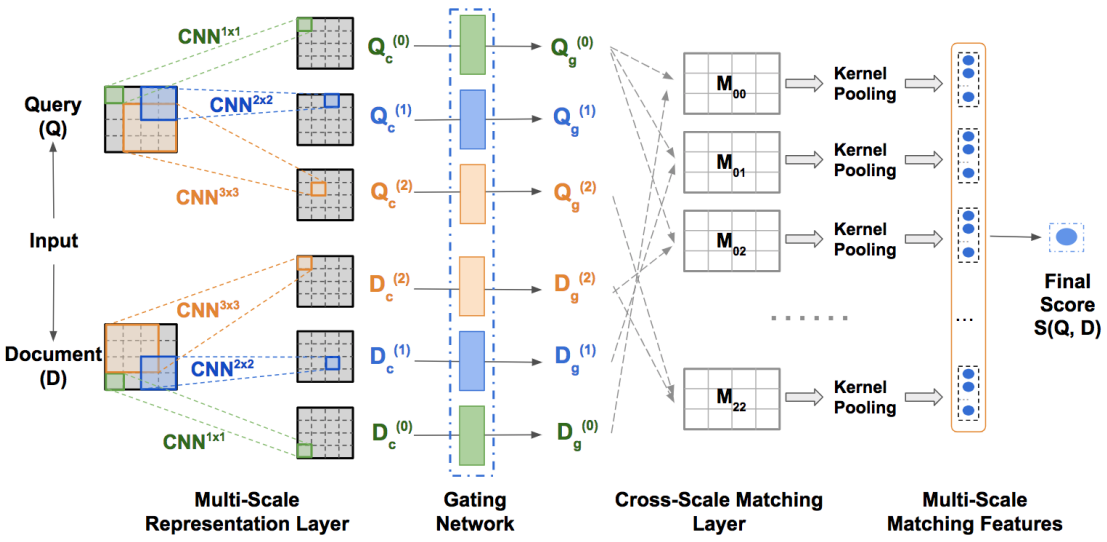


Figure 5.1: The DISH-Conv architecture. The input is the local convolution feature vectors of a query and a document. The multi-scale representation layer uses spatial convolutions of different kernel sizes to generate features at multiple scales. The gating network computes the importance of each feature, and suppresses features from unimportant areas. The cross-scale matching layer compares the similarity between query regions and document diagram regions of different sizes, and forms the similarity matrices for different scales. The kernel pooling aggregates the similarity matrices into cross-scale matching features. A *learning-to-rank* layer combines all features and generate a final score.

5.3 Conv-KNRM to DISH-Conv: Local Convolutional Matching Network

DISH-HP is designed to be simple and efficient in order to be used for initial ranking. The simple architecture has several drawbacks that may limit its search effectiveness. First, DISH-HP uses several heuristics, such as which bins are used and how low-density areas are detected. These heuristics may not be optimal. More importantly, the local features are only for a fixed size of image regions (e.g. 16 pixel $\times$ 16 pixel), while in many cases the same part may be drawn at different scales in different diagrams.

This section proposes DISH-Conv, a *Local Convolutional Matching Network* for engineering diagram search. The drawbacks of DISH-HP are addressed by using machine-learned weights to replace heuristics and using better network architectures to generate richer features.

DISH-Conv is inspired by Conv-KNRM (Chapter 4). It adapts the use of convolutions and cross-matching from text search to diagram search. The architecture of DISH-Conv is shown in Figure 5.1. Given local convolution features from a query and a document diagram, it builds multi-scale region representations through spatial convolutions, generates cross-scale local match signals through the cross-matching layer. It then ranks diagrams with the cross-scale match signals using kernel-pooling and learning-to-rank. A novel gating network is developed that mutes unimportant region features in the query and the document.

5.3.1 Generating Multi-Scale Representations

DISH-Conv starts with the same process as DISH-HP. Local feature vectors are extracted from the query diagram Q and the document diagram D. The feature vectors are organized in a 3D array of size $[H, W, d]$: $H \times W$ local regions each with a d dimension feature vector.

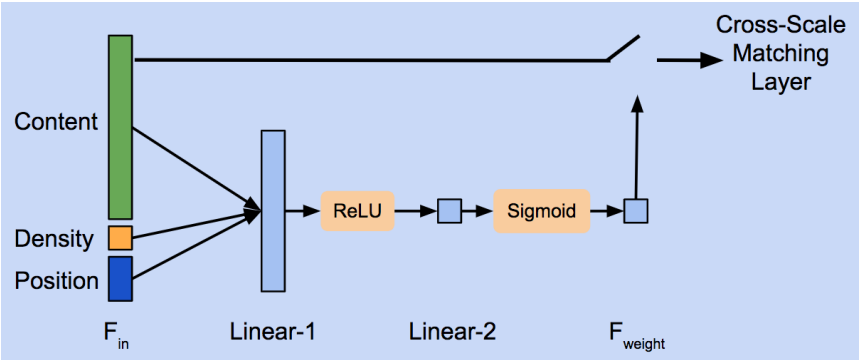


Figure 5.2: The gating network architecture of DISH-Conv.

Intuitively, features extracted from regions of different sizes could improve the robustness on diagrams of different scales. Following this intuition, DISH-Conv uses 2D spatial convolutions to generate features for larger regions. A 2×2 kernel builds representations from 4 adjacent regions, generating a new feature vector whose receptive fields is 4 times larger than the original feature vector. Similarly, 3×3 kernels generate representations for even larger regions. We also employ a 1×1 convolution. It has the same receptive fields as the input features, but can transform the features to better fit the ranking task.

The convolutions form a pyramid of region representations at multiple scale levels. In addition to scale, the convolution can also learn other types of transformation. It is able to model rotation because rotation changes are essentially a linear transformation of the pixel coordinates. It can also learn transformation in texture and structure, optimizing the input features toward the specific task and dataset.

5.3.2 Gating Network

During the development of DISH-HP, we identified that the model should ignore unimportant regions of a diagram. DISH-HP employed a heuristic that compares the L2-norm of local feature vectors to a fixed threshold. However, it is not guaranteed that the features of blank areas are always close to zero; and, some high-density areas are also unimportant, for example, callouts and bubbles.

A gating network was developed to automatically learn the importance of local features and to down-weight unimportant features during the matching process (Figure 5.2). The gating network predicts the importance of a region based on three types of signals: content, position, and density. The content is represented by the region emebedding from the multi-scale representation layer (C features, C is the number of convolution channels). The position features are the coordinates of the center of the perceptive field (2 features). The density feature is the L2-norm as used in DISH-HP (1 feature). These signals are concatenated, generating a $C + 3$ dimension input vector F_{in}, and passed through a 2-layer Perceptron (Nilsson, 1965):

$$\vec{h} = ReLU(W_1 \vec{F}_{in} + B_1) \tag{5.3}$$

$$F_{weight} = \sigma(W_2 \vec{h} + B_2), \tag{5.4}$$

where W_1, B_1, W_2, and B_2 are the trainable parameters of the Perceptron. σ is the Sigmoid function that converts the free-range network output into the range of [0, 1]. The output, $F_{weight} \in [0, 1]$, is multiplied to the original feature vector. F_{weight} close to 1 means that the local feature vector is important and is

allowed to pass through the gate; F_{weight} close to 0 means that the feature vector is unimportant and its effect on the ranker should be muted.

The gating network is trained jointly with the other parts of DISH-Conv, so that it automatically learns to recognize regions that are important for the search task. The importance of a local feature vector is based on not only the density, but more importantly, its content. It allows dense but meaningless regions, such as bubbles, to be detected.

5.3.3 Ranking with Cross-Scale Match Signals

The multi-scale local region representations, with unimportant regions suppressed by the gating network, are sent to the next cross-scale matching layer. This layer calculates the similarity between each pair of query-document regions. The regions can be taken from different scales. For example, a 16×16 pixel area in the query diagram is compared to 16×16, 32×32, and 48×48 areas in the candidate diagrams (from 1×1, 2×2, and 3×3 convolution). In total, the cross-scale match is performed among 9 different scales.

For each of the 9 scale combination, the match signals are aggregated through *kernel-pooling* (Xiong et al., 2017b). Kernel-pooling is a soft version of the histogram pooling used in DISH-HP that approximates the histogram function with RBF kernels. It makes the histogram pooling differentiable so that gradients can be back-propagated to the previous convolution layer. The resulting kernel features are similar to the histogram features in DISH-HP. The kernel features from all 9 scale combinations are then combined by a learning-to-rank layer to generate the final score. The learning-to-rank layer assigns different importance to local match signals with different strength (e.g. 'strongly similar', 'weakly similar', 'dissimilar') and at different scales (e.g. 'small query region to small document region' or 'large query region to medium document region'). The output is the predicted relevance score between the query and the document $s(Q, D)$.

DISH-Conv is trained to minimize a standard pairwise loss function:

$$L(Q, P, N) = max(0, \alpha - s(Q, P) + s(Q, N)). \tag{5.5}$$

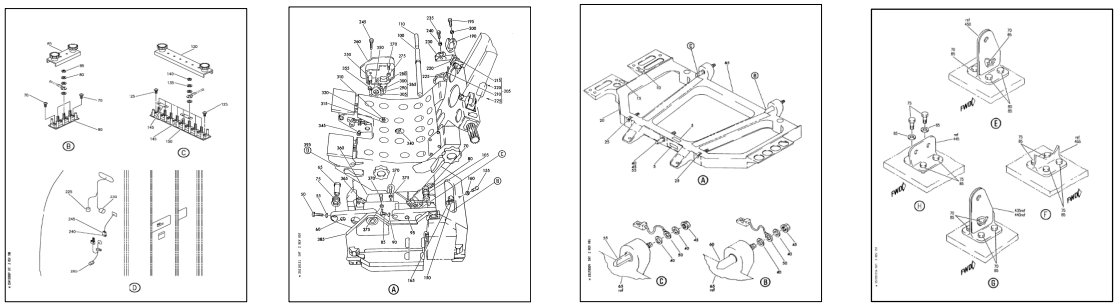
The loss function keeps relevant diagrams P closer than any negative diagrams N for each query Q, with a margin α. A query's relevant documents are taken as positive examples. Negative examples are sampled uniformly from the dataset.

To summarize, DISH-Conv used machine-learned convolutions and cross-scale match to hand scale and rotaion change. A novel gating network is proposed to suppress unimportant regions in the query and the document. Collectively, these elements address several issues identified in DISH-HP, and should enable DISH-Conv to produce more accurate rankings.

5.3.4 Relation to Text-Ranking Models

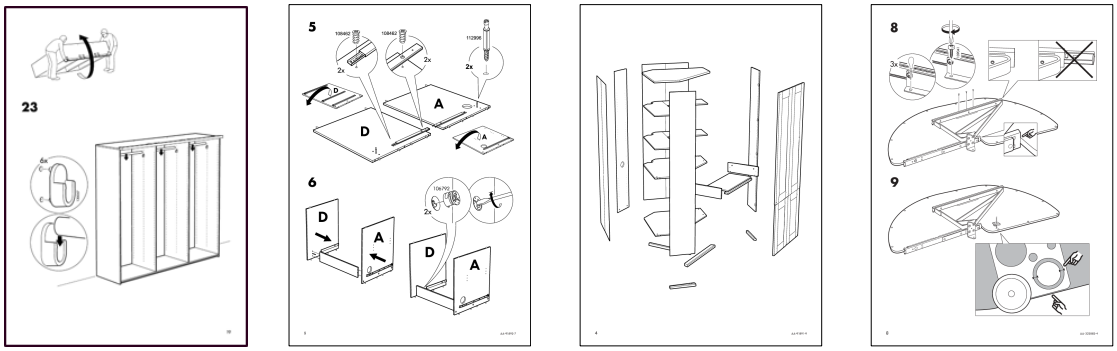
DISH-HP and DISH-Conv are based on the interaction-based neural ranking models designed originally to model query-document interactions in text search (Guo et al., 2016a; Xiong et al., 2017b; Dai et al., 2018). They are expected to retrieve more-relevant documents than global matching methods because the local, region-level matching provides more evidence than matching at the global image level.

DISH-Conv additionally used the convolution and cross-matching techniques from Conv-KNRM. But, the the intuition of using the convolutions and cross-matching is slightly different from Conv-KNRM. Here, our focus is on *scales* and *rotations*. We employ spatial convolutions to learn feature transformations that can tolerate scale and rotation changes, and use cross-matching to generate match signals across



(a) (b) (c) (d)

Figure 5.3: Diagrams from the Boeing dataset.



(a) (b) (c) (d)

Figure 5.4: Diagrams from the Ikea dataset.

multiple scales. We also developed a novel gating network to learn local region importance from position, density, and more importantly, content.

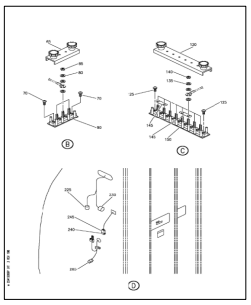
5.4 Experimental Methodology

This section introduces first two datasets used for experimental evaluation: the Boeing dataset and the Ikea dataset.

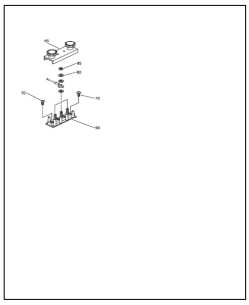
5.4.1 The Boeing Dataset

The Boeing dataset was provided by the Boeing Company. It consists of $12,536$ engineering diagrams for Boeing 777 interiors. See Figure 5.3 for a few examples.

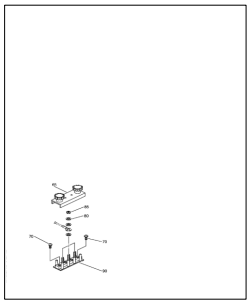
The original dataset did not contain queries with relevance assessments. We developed an automated query generator to create a training and evaluation data. It tries to locate parts in a diagram that would make reasonable queries. The query generator first uses the DBSCAN algorithm (Ester et al., 1996) to cluster the pixels in the image. After the clusters are found, the query generator finds the bounding box of each cluster and filters out clusters that are large or too small. The remaining clusters are cropped from the image to form new query images. A further filtering process is then applied to remove locators – a location indicator that appears in almost all diagrams.



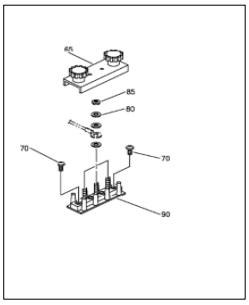
(a) A relevant document.



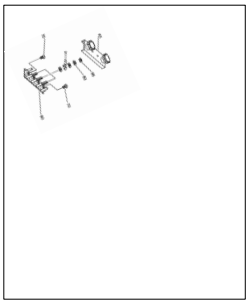
(b) Query with no change (psr).



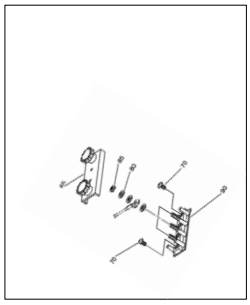
(c) Query with changed position (Psr).



(d) Query with changed scale (pSr).

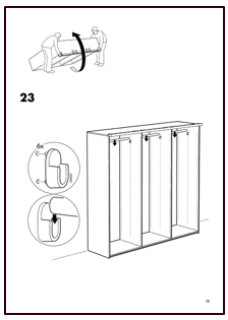


(e) Query with changed rotation (psR).

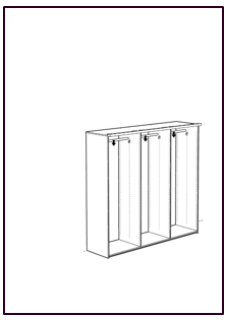


(f) Query with changed position, scale, and rotation (PSR).

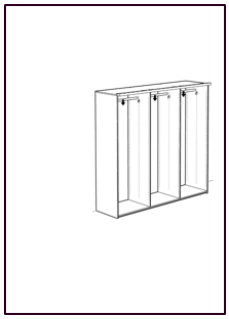
Figure 5.5: A document, a query, and four automatically-generated query variants for the Boeing dataset.



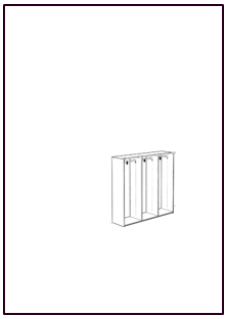
(a) A relevant document.



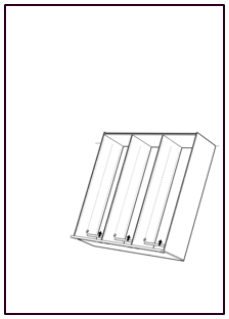
(b) Query with no change (psr).



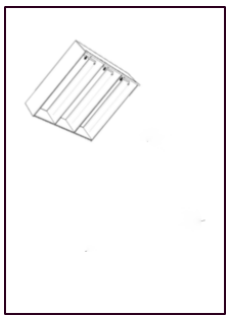
(c) Query with changed position (Psr).



(d) Query with changed scale (pSr).



(e) Query with changed rotation (psR).



(f) Query with changed position, scale, and rotation (PSR).

Figure 5.6: A document, a query, and four automatically-generated query variants for the Ikea dataset.

In typical use, queries would not have the same position, scale, and rotation as a region of a relevant diagram. The initial set of queries is named *psr* queries, with lower case letters indicating that the position (p), scale (s), and rotation (r) of the query image are unchanged. Four additional sets of queries were generated to have specific characteristics and increasing difficulty for use in diagnosing weaknesses in retrieval algorithms. The names of these query sets use capitalized letters to mark relative changes. *Psr* queries change the position of the query image, but keep scale and rotation invariant. *pSr* queries change the scale of the query image within the range $[0.5, 2]$. *psR* queries change the rotation 0 to 360 degrees. *PSR* queries change position, scale, and rotation simultaneously. Figures 5.5 show an example of a document image, a query automatically cropped from the document, and the position/scale/rotation/all-changed queries.

When generating queries, a position, scale, or rotation change is generated randomly within an appropriate range. The transformed query is required to fit entirely within the image boundary so that it does not lose information during the transformation. For position and scale changes, simple rules ensure that the transformed query lies within the image boundary. For rotation changes, it is more complex to predict the part boundaries. Therefore, if the rotated query was not entirely within the image boundaries, the change was discarded, and a new random rotation angle generated. If the rotation change failed five times, the image was discarded from the set of queries. Thus, the *psr*, *Psr*, and *pSr* query sets all have the same number of queries (4,458), but the *psR* and *PSR* query sets are somewhat smaller [2]. Table 5.1 shows the characteristics of the Boeing collection and queries.

In the automated query generation process, the source diagram is treated as the query's (only) relevant document. A good retrieval system should rank that document highly. Mean Reciprocal Rank (MRR) is a common metric when there is one relevant document. We also want to know the number of queries for which the relevant document appears at rank 1 and in the top 10 results, thus we also use Recall at cutoff at 1 and 10 (R@1 and R@10).

5.4.2 The Ikea Dataset

The Boeing dataset is restricted to be accessed on Boeing machines. The unavailability to export data and the difficulty installing software on Boeing machines makes it difficult to compare our models to other baselines on the Boeing dataset. Therefore, we created a new dataset based on Ikea furniture assembly manuals was developed to support experimental evaluation. Although we are unable to redistribute the dataset, others can create a substantially similar dataset based on the descriptions below and additional information posted on our website[3]. *The Ikea dataset is our main evaluation set.*

We downloaded product assembly manuals from Ikea [4]. Each manual was split into page images. Images that were entirely blank, primarily text, or duplicates of other images were discarded. The result was a corpus of 13,464 furniture assembly diagrams. Each diagram was a black-and-white pdf document. See Figure 5.4 for some examples.

Same as the Boeing dataset, queries were generated automatically by cutting out regions of a diagram and applying position, rotation, and/or scale changes to approximate the difficulty of real queries. 5,000 regions were cropped using the automated query generator. After random changes to position, scale, and/or rotation, there were 4,458 position changed queries (Psr), 3,649 scale changed queries (pSr), 3,649 rotation changed queries (psR), and around 300 queries with position, scale, and rotation changed. Fig-

[2]Randomly changing position, scale, and/or rotation occasionally fails to generate a query, for example, when a part is rotated out of frame. If a transformer fails to produce a valid query after five attempts, the query is discarded. Thus, 5,000 cropped regions doesn't produce 5,000 queries.

[3]`http://boston.lti.cs.cmu.edu/appendices/TheWebConf2019-Zhuyun-Dai/`

[4]`https://www.ikea.com/ms/en_US/customer_service/assembly_instructions.html`

Table 5.1: Characteristics of the Boeing Dataset and the Ikea Dataset

	Description	Boeing	Ikea
Corpus	Diagrams from Boeing / from Ikea manuals	12,536	13,464
psr	No change	4,458	3,649
Psr	Random position change	4,458	3,649
pSr	Random scale change (0.5 to 2)	4,458	3,649
psR	Random rotation change (0 to 360)	3,274	2,988
PSR	Position, scale, & rotation change	3,638	3,005

ure 5.6 illustrates the queries. Evaluation used MRR and Recall on the automatically generated queries. A small set of *manual* labeled queries with multiple relevant documents is studied in Section 5.5.5.

5.4.3 Proposed Models and Baselines

Proposed models include two local matching networks, `DISH-HP` and `DISH-Conv`. `DISH-HP` is the unsupervised ranking algorithm proposed in Section 5.2. `DISH-Conv` is the supervised local convolutional matching network proposed in Section 5.3. The proposed models are compared to several baselines, including a bag-of-words image retrieval model based on SIFT features, a neural image retrieval model based on CNN features, and an encoder-decoder network designed specifically for line drawings.

`LIRE` (Lux and Marques, 2013) is an open source image retrieval system based on Lucene. It uses bag-of-words image retrieval models with SIFT features. The indexing and retrieval are unsupervised. We used the open source software [5] to index the images in our corpus and perform the retrieval.

`FG-SBIR` (Pang et al., 2017a) is a neural retrieval system designed for sketches and line drawings. It is based on an encoder-decoder architecture. The encoder is based on the VGG-16 network (Simonyan and Zisserman, 2014); the decoder consists of a chain of upsampling blocks to reconstruct the sketch image from the VGG-16encoder's fully connected layer, helping the encoder to learn a richer representation. The output from the encoder is taken as the image descriptor and used for retrieval. We used the open source implementation [6]. `FG-SBIR` is a *global* matching approach as the entire image is encoded into a single embedding; local region features are not preserved.

`CAM` (Jimenez et al., 2017) is a state-of-the-art instance retrieval approach based on deep convolutional neural networks. It uses CNN features extracted from an off-the-shelf VGG convolution network trained on ImageNet (Deng et al., 2009). For each pre-defined class in ImageNet, `CAM` generate a class-focused image vector by weighted-summing the CNN features based on their activation to this class. The vectors from all classes are then normalized, whitened, and sum-pooled into a single image vector. Image retrieval is based on the Euclidean distance between the final image vectors. We took the pre-trained `CAM` model released by the authors [7]. We did not re-train `CAM` on the Ikea dataset due to lack of class labels. `CAM` is also a *global* matching approach as local region features are not preserved

The above three baselines used off-the-shelf software. We do not have control over how they process data and extract features. To gain better understanding of the proposed models, we created a set of `DISH` variants as additional controlled baselines. They differ from `DISH-HP` and `DISH-Conv` only in the architectures of the matching networks. The implementation framework, data processing, and feature extraction are controlled to be exactly the same.

[5] http://www.lire-project.net/
[6] https://github.com/zoeyangdw/FG-SBIR
[7] https://github.com/imatge-upc/retrieval-2017-cam

DISH-SPoC and DISH-MPoC are two global variants of DISH-HP. They were created to test the effectiveness of local matching against global matching. DISH-SPoC is based on the SPoC descriptor proposed by Babenko and Lempitsky (2015), but uses the DISH framework. It takes the same CNN features as DISH-HP, and applies the same filtering on white areas. The filtered CNN features are then aggregated through sum-pooling to generate a single, global image feature vector. The similarity between the query and a document diagram is calculated from the cosine similarity between the corresponding vectors. DISH-MPoC is a similar method that replaces the sum-pooling with max-pooling. DISH-SPoC and DISH-MPoC are unsupervised methods.

DISH-Conv-N is DISH-Conv without the spatial convolution; the letter N stands for 'No convolution'. In DISH-Conv-N, the input local CNN features are directly used to calculate region similarities. The model lacks the multi-scale region representation and cross-scale matching in DISH-Conv. DISH-Conv-N can also be viewed as a supervised version of DISH-HP with the hard histogram replaced by soft RBF kernels. DISH-Conv-N is a supervised model.

DISH-Conv-S is a simplified version of DISH-Conv that replaces the spatial convolution with spatial pooling when generating multi-scale region representations. The spatial pooling takes the average of features from smaller regions to generate features for larger regions. Pooling does not have additional parameters, so this model is faster and easier to tune compared to DISH-Conv. DISH-Conv-S separates the effects of convolution from the effects of aggregating small regions into larger regions. It is also a supervised model

We employed a two-stage search pipeline. In the first stage, an initial ranker takes the query and compares it against all diagrams in the corpus, and returns a list of similar diagrams as candidates. In the second stage, a different ranker re-ranks the top 2,000 candidates. DISH-HP, DISH-SPoC and DISH-MPoC are simple models designed for efficient ranking; they were tested on the initial ranking task. DISH-Conv, DISH-Conv-N, and DISH-Conv-S were designed to improve DISH-HP with more complex models; they were tested on the re-ranking task. LIRE, FG-SBIR, and CAM were tested in both stages, providing baselines for both tasks.

5.4.4 Model Configuration and Implementation

Off-the-shelf baselines (LIRE, FG-SBIR, and CAM) used their own implementations for feature extraction. All DISH models took the features from an off-the-shelf VGG-16 network trained on ImageNet[8]. Following prior research (Razavian et al., 2014), we extracted features from the last convolution layer. This layer divides the image into $R = W \times H = 14 \times 14 = 196$ grids, each corresponding to a 16 pixel $\times$ 16 pixel rectangular region in the diagram. For each of the 196 regions, a $d = 512$ dimension feature vector is generated. We did not fine-tune the VGG-16 network on engineering diagrams, as the focus of this work is on modeling the *interactions* between local features.

Supervised models (DISH-Conv, DISH-Conv-N, and DISH-Conv-S) were trained and tested via 5 fold cross-validation. For each query we sample 10 negative examples. We used the Adam optimizer with batch size 16, learning rate 0.001, and a learning rate decay of 0.5 when validation loss does not decrease. The training stops after 30 epochs or when learning rate drops below $3.12e-5$. The models were implemented in PyTorch.

For DISH-HP, the top $K = 2$ bins were used and low-density area filter threshold T was set as 30. The parameters were chosen through a simple parameter sweep. The kernel pooling layers in DISH-Conv, DISH-Conv-N, and DISH-Conv-S all followed the configurations suggested by prior research (Xiong et al., 2017b) and used 11 kernels. The first kernel is the exact match kernel where $\mu = 1, \sigma = 10^{-3}$,

[8] https://download.pytorch.org/models/vgg16-397923af.pth

Table 5.2: Initial ranking accuracy on several types of queries. The query is compared to all document diagrams in the Ikea datset.

	Invariant (psr)			Position (Psr)			Scale (pSr)			Rotation (psR)			ALL (PSR)		
Method	MRR	R@1	@10	MRR	R@1	R@10	MRR	R@1	R@10	MRR	R@1	R@10	MRR	R@1	R@10
LIRE	0.19	14%	29%	0.17	9%	19%	0.03	1%	3%	0.00	0%	1%	0.00	0%	0%
CAM	0.38	6%	48%	0.31	6%	41%	0.23	5%	32%	0.08	2%	13%	0.04	2%	7%
FG-SBIR	0.44	23%	36%	0.41	19%	35%	0.39	19%	32%	0.10	5%	14%	0.06	3%	8%
DISH-SPoC	0.52	45%	63%	0.48	41%	59%	0.37	30%	48%	0.10	7%	16%	0.07	4%	11%
DISH-MPoC	0.66	58%	76%	0.57	50%	70%	0.44	37%	57%	0.12	9%	18%	0.08	5%	12%
DISH-HP	**0.68**	**59%**	**84%**	**0.65**	**56%**	**82%**	**0.54**	**44%**	**70%**	**0.15**	**11%**	**21%**	**0.10**	**7%**	**14%**

Table 5.3: Re-ranking accuracy on several types of queries. The top 2,000 documents retrieved from the Ikea dataset by DISH-HP were re-ranked by each method.

	Invariant (psr)			Position (Psr)			Scale (pSr)			Rotation (psR)			ALL (PSR)		
Method	MRR	R@1	R@10	MRR	R@1	R@10	MRR	R@1	R@10	MRR	R@1	R@10	MRR	R@1	R@10
DISH-HP	0.68	59%	84%	0.65	56%	82%	0.54	44%	70%	0.15	11%	21%	0.10	7%	14%
LIRE	0.55	23%	87%	0.66	27%	91%	0.13	2%	46%	0.01	0%	1%	0.01	0%	0%
CAM	0.76	60%	88%	0.68	58%	83%	0.60	47%	71%	0.18	12%	22%	0.11	8%	15%
FG-SBIR	0.82	72%	89%	0.76	70%	84%	0.64	58%	72%	0.20	14%	23%	0.11	7%	14%
DISH-Conv-N	0.93	89%	99%	0.83	83%	98%	0.73	65%	85%	0.21	15%	30%	0.09	6%	14%
DISH-Conv-S	**0.94**	**89%**	**99%**	**0.90**	**84%**	**98%**	0.74	65%	86%	0.22	17%	31%	0.10	6%	16%
DISH-Conv	0.91	87%	97%	0.82	74%	94%	**0.78**	**70%**	**90%**	**0.38**	**28%**	**57%**	**0.17**	**10%**	**30%**

or bin $[1, 1]$. The other 10 kernels equally split the cosine range $[-1, 1]$: the μ of bin centers were: $\mu_1 = 0.9, \mu_2 = 0.7, ..., \mu_{10} = -0.9$. The σ of the soft match bins were set to be 0.1. The multi-scale presentation layer in DISH-Conv uses three types of spatial convolutions with different kernel sizes $(1 \times 1, 2 \times 2, 3 \times 3)$, 128 output channels, and a stride of 1.

5.5 Experimental Results

Due to the difficulty in installing baseline softwares on the Boeing machines and the inability to export Boeing data, we mainly use the Ikea dataset for evaluation.

Two experiments on the Ikea dataset tested the accuracy of the different methods of retrieving diagrams in ranking and re-ranking configurations on large-scale auto-generated relevance labels. In addition, we report the experimental results of DISH methods on the Boeing dataset.

Two analyses investigate the behavior and effectiveness of the convolutions and the gating network. An evaluation on a small set of manually-judged queries was conducted to confirm the results from experiments with auto-generated relevance labels.

5.5.1 Initial Ranking Performance of DISH-HP

DISH-HP is simple, fast, and does not require training. It was tested on the first stage ranking task where the query is compared to all diagrams in the collection. As shown in Table 5.2, DISH-HP outperforms all other methods by large margins.

Among the baselines, only LIRE uses SIFT features; the rest use deep CNN features. LIRE is the least accurate, demonstrating the effectiveness of deep CNN features on engineering diagrams (Table 5.2).

FG-SBIR, which is designed for sketches and line drawings, has higher precision (MRR and Recall@1), but CAM retrieves more relevant diagrams (Recall@10). These results indicate that CNN feature extractors learned from photographs can also extract useful patterns from engineering diagrams.

DISH-SPoC and DISH-MPoC outperforms FB-SBIR and CAM. These methods are all representation-based methods based on CNN features. The main differences of DISH-SPoC and DISH-MPoC with FG-SBIR and CAM are i) they have low-density region filtering, and ii) they use simple sum/max to aggregate local CNN features while FG-SBIR and CAM learn feature aggregation from a dataset. Low-density region filtering is important because these regions bring in noisy signals. The simple pooling functions are task-neutral, while FG-SBIR and CAM are tuned for their training datasets. DISH-MPoC is the strongest baseline.

DISH-HP outperforms all other methods by large margins. Like DISH-MPoC, DISH-HP uses deep CNN features and filters blank regions. However, DISH-HP is an interaction-based method that leverages local region matching evidence. In representation-based methods such as DISH-MPoC, matching is at the entire image level; no regional similarity signals are used. They squeeze the image into a 512 dimensional vector. The vector is effective at capturing high-level visual features, such as the shape of a part. However, it loses information at lower granularity levels. As a result, irrelevant diagrams are retrieved, for example, a wheel is retrieved for a round washer because they have similar shapes. Representation-based methods also missed documents that are partially matched to the query because the other parts in the diagram make the feature vector less similar to the query's feature vector. On the other hand, DISH-HP leverages local interaction signals that tell if the sub-parts between two diagrams are similar. It helps to filter out irrelevant documents and to retrieve partially matched documents.

Table 5.2 also shows DISH-HP's sensitivity to position, scale, and rotation changes. DISH-HP is robust to position changes. It had almost the same performance on the position changed queries (Psr) compared to the invariant queries (psr), whereas DISH-MPoC suffered a significant drop when the query is shifted. DISH-HP was relatively sensitive to scale changes: the Recall@10 decreased from 84% to 70% when the query scale was changed (pSr). Rotation changes were more difficult: the Recall@10 was only 21% on rotation changed queries (psR). Queries that simultaneously changed position, scale and rotation (PSR) were the most difficult.

In summary, the experimental results on the baselines show the effectiveness of deep CNN features on engineering diagrams compared to hand-crafted SIFT features. A comparison between models learned on ImageNet and models learned on line drawings indicate that off-the-shelf CNN feature extractors can achieve good Recall, although tuning them for line drawings may increase Precision. Experimental results on DISH-HP demonstrate the power of *interaction-based* approaches. The local region interaction signals provide evidence to filter out irrelevant diagrams that share similar shapes with the query but differ in detail. It also enables retrieving relevant diagrams that partially match the query. DISH-HP largely outperforms commonly-used kNN-style representation-based approaches, including state-of-the-art instance retrieval models. The Recall of DISH-HP was not perfect, especially on scale and rotation changes. But it may serve as good starting point for a retrieval pipeline and to generate training data for more advanced ranking models.

5.5.2 Re-ranking Performance of DISH-Conv

The second experiment investigated DISH-Conv and its variants. DISH-Conv uses learned weights, multi-scale matching, and convolutional features to improve ranking, especially for queries at different scales and rotations. Due to its greater computational cost, DISH-Conv was only tested as a second stage that re-ranks the top N results returned by a first-stage ranker such as DISH-HP. As a reference, we also tested the baseline methods on the re-ranking task.

As shown in Table 5.3, LIRE failed to improve the initial ranking of DISH-HP, but both CAM and FG-SBIR were able to improve it. DISH-Conv greatly improved ranking accuracy. In invariant (psr) and position changed queries, over 80% of the relevant documents were ranked first. On queries with scale (pSr) changes, DISH-Conv is able to find 90% of the relevant documents by rank 10, 28% more than DISH-HP. On queries with scale (pSr) changes, DISH-Conv almost tripled Recall@10 of DISH-HP. Together, they lead to higher performance on the most difficult PSR queries.

The two variants, DISH-Conv-N and DISH-Conv-S, help us to understand the sources of effectiveness.

DISH-Conv-N is DISH-Conv without convolution. It can be viewed as a supervised version of DISH-HP , where the hard histogram is replaced by soft RBF kernels, the weight of the kernels is learned, and the heuristic-based region filter is replaced by an end-to-end trained gating network. DISH-Conv-N outperforms DISH-HP, demonstrating that machine-learned kernel weights and machine-learned gating network are more effective than the heuristics.

DISH-Conv-S uses spatial pooling to generate multi-scale representations and matches two diagrams at multiple scale levels. Compared to DISH-Conv-N that only models single-scale match, the pooling-based multi-scale signals in DISH-Conv-S slightly improved the search accuracy for scale changed queries (pSr). Unexpectedly, the model has very high accuracy on position changes (Psr). When a query changes position, the perceptive field receives different content, and the convolution features changes. Pooling features from adjacent regions produces a descriptor for a larger region and compensates for the position change. The results shows that matching at multiple scales is not only effective for scale invariance, but also improves position invariance.

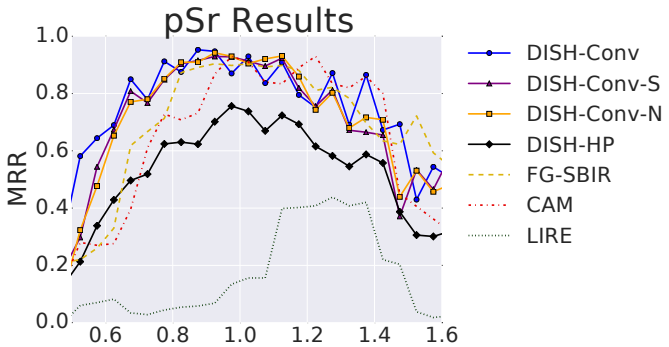
DISH-Conv is the most effective model on queries with scale and rotation changes (pSr, psR and PSR). In contrast to DISH-Conv-S which uses simple average pooling to generate features, DISH-Conv learns a matrix transformation of small-region features to larger-region features. The transformation is more powerful in handling complex changes in scale and rotation. Besides, the parameters are *learned* from the training data, so that they are tuned to fit the characteristics of the diagram dataset and the ranking task. The convolution, the cross-scale matching, together with machine-learned weights, make DISH-Conv more accurate and more robust.

5.5.3 Scale and Rotation Invariance Analysis

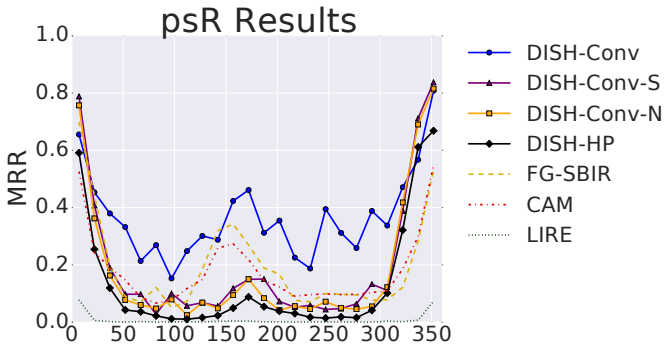
DISH-Conv uses convolutions and cross-matching to handle scale and rotation differences between queries and diagrams. An analysis on the Ikea dataset investigates the effectiveness of these components (Figure 5.7).

Figure 5.7a shows each model's sensitivity to scale changes in pSr queries. Most models are best when the scale is unchanged (1.0). Ranking accuracy gradually decreases when the query is scaled down or up. DISH-Conv and DISH-Conv-S are best for all scale changes, due to their multi-scale representations and cross-scale matching. DISH-Conv is more robust to extreme scale changes, demonstrating that convolution is more powerful than pooling.

Figure 5.7b shows each model's sensitivity to rotation changes in psR queries. All models except DISH-Conv have a bow-like curve. Their MRR scores are high when there is no rotation, but drop to near 0 when the query is rotated $\pm$ 50 degrees. MRR improves somewhat near 180 degrees rotation because some queries are symmetric (e.g., rectangles); the symmetry effect is more obvious on FG-SBIR and CAM. Among all models, DISH-HP is the least able to handle rotation, and DISH-Conv is the most robust, although there is still a significant impact. Convolutions in DISH-Conv learn feature transformations that compensate somewhat for the rotation changes.



(a) Scale change



(b) Rotation change

Figure 5.7: Re-ranking accuracy (MRR) of `DISH-Conv` for different amounts of change in query scale and rotation on the Ikea dataset.

5.5.4 Gating Network Analysis

`DISH-Conv` uses a gating network that estimates the importance of each region based on three types of features: the region's pixel density, estimated by the L2 norm of the feature vector; the region's position, represented by the coordinates of the region center; and the region's content, represented by its feature vector. An ablation study on the Ikea dataset examined the effectiveness of the gating network. Figure 5.8 compares the accuracy of `DISH-Conv` with and without the gating network. It also compares gating networks using different types of features.

As shown in Figure 5.8, the MRR of `DISH-Conv` drops when the gating network is disabled. Without the gating network, the matches from meaningless areas dominated `DISH-Conv`'s match signals, which produces poor results.

Comparison of gates using position, density, and/or content features indicates that content is the most powerful feature. The content gate learns the importance of a region from its CNN feature vectors, essentially capturing the *meaning* of the region. The position gate predicts a region's importance based on its location in the diagram, assuming that the center of the image is more likely to contain content of interest, which is not necessarily true. Using the position feature alone leads to worse results. The density gate used the L2 norm of the region feature vector, which is similar to the low-density region filter used by `DISH-HP`. The density gate is moderately better than having no gate. Combining all three types of features (`full`) provides further improvement, but the best combination omits the density feature, which is subsumed by the content feature.

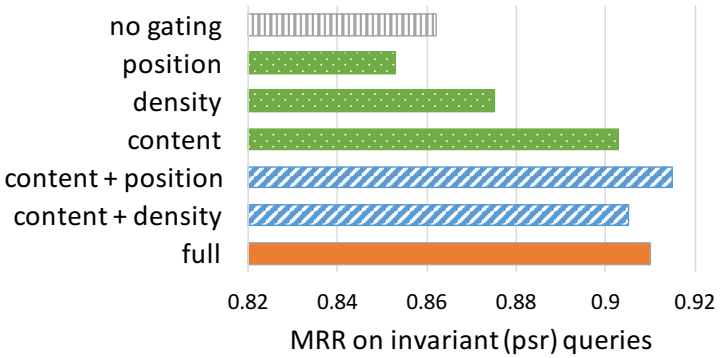


Figure 5.8: Re-ranking accuracy (MRR) of DISH-Conv on invarinat (psr) queries when gating uses different features. Dataset: The Ikea dataset.

Table 5.4: Ikea dataset manual relevance assessments statistics.

	Relevance Judgment Criteria	Avg Doc/Query
3	Same furniture	5.7
2	Likely to be the same furniture	1.3
1	Different, but similar assembly	5.8
0	Not related	63.6

In summary, identifying important regions and ignoring noisy regions is crucial for matching diagrams. A novel gating network was proposed to detect regions that are unimportant for ranking. Experimental results show that the gating network boosts the ranking accuracy of Local-Conv. The feature ablation study shows that learning to recognize unimportant contents delivers most of the gain, followed by the position of the region within the image.

5.5.5 Ikea Manual Relevance Assessments

The previous experiments measure search accuracy using auto-generated relevance labels. This experiment measures accuracy using a small set of manual relevance judgments.

For each of 50 queries, the top 30 diagrams retrieved by Psr, pSr, and psR query variants were judged. 76 diagrams per query were assessed, on average. Three assessors judged the relevance of each diagram on a scale of $0-3$. Table 5.4 shows the relevance criteria and distribution. Diagrams with labels of 2 and 3 were considered relevant. On average, a query had 7 relevant diagrams.

These assessments were used to evaluate DISH methods and the best baseline FG-SBIR in the re-ranking setting. We report results for rotation (psR) queries because they differentiate the ranking methods the most. As shown in Table 5.5, the relative order of the ranking methods using manual labels is consistent with that on automatically generated labels: all DISH-Conv methods improved the initial ranking from DISH-HP; DISH-Conv methods were better than FG-SBIR; and DISH-Conv was most accurate. R@1 and R@10 values are lower for manual labels than for automatic labels because there are 7$\times$ more relevant documents in this condition. MRR values are higher for manual labels, meaning that some top-ranked documents that are considered non-relevant by the automatic labels are actually relevant.

In summary, the relative order of the ranking methods does not change when switching from auto labels to manual labels, validating the conclusions drawn from previous experiments.

Table 5.5: Re-ranking accuracy on 50 rotation queries (psR) with manual relevance assessments on the Ikea dataset.

	Automatic Labels			Manual Labels		
Method	MRR	R@1	R@10	MRR	R@1	R@10
DISH-HP	0.11	6%	18%	0.17	3%	10%
FG-SBIR	0.13	7%	21%	0.16	4%	10%
DISH-Conv-N	0.15	8%	28%	0.23	6%	14%
DISH-Conv-S	0.15	10%	22%	0.19	5%	11%
DISH-Conv	**0.31**	**22%**	**46%**	**0.37**	**8%**	**22%**

Table 5.6: Accuracy of DISH-HP and DISH-Conv on the Boeing dataset. The top 2,000 documents retrieved from the Boeing dataset by DISH-HP were re-ranked by DISH-Conv.

	Invariant (psr)			Position (Psr)			Scale (pSr)			Rotation (psR)		
	MRR	R@1	R@10	MRR	R@1	R@10	MRR	R@1	R@10	MRR	R@1	R@10
DISH-HP	0.16	12.6%	23.5%	0.07	4.4%	12.6%	0.05	3.0%	9.8%	0.01	0.1%	2.4%
DISH-Conv-N	0.33	26.4%	43.3%	0.21	15.3%	33.1%	0.17	11.1%	27.9%	0.02	1.0%	3.4%
DISH-Conv-N	0.15	10.5%	21.7%	0.13	9.5%	14.8%	0.12	8.2%	19.5%	0.01	0.4%	2.2%
DISH-Conv	0.54	45.7%	68.6%	0.35	27.1%	50.6%	0.30	22.7%	45.1%	0.06	2.6%	11.2%

5.5.6 Results on the Boeing Dataset

This last experiment evaluates the DISH methods on the Boeing dataset. Due to inability to install software for the baselines on the Boeing machines, this experiment only compared DISH variants. Table 5.6 shows the results. The evaluation did not include evaluation on ALL change queries (PSR) as the recall was very low on PSR queries and does not reflect the differences between re-ranking methods.

The relative order of the ranking methods on the Boeing dataset is same as that on the Ikea dataset: all DISH-Conv methods improved the initial ranking from DISH-HP, and DISH-Conv was most accurate. The absolute accuracy on the Boeing dataset was much lower than on the Ikea dataset (Table 5.2 and 5.3), revealing that Boeing diagrams are more difficult. Importantly, the rotation changes in query images have large impacts on the accuracy of the Boeing dataset, indicating that the convolutions used in DISH-Conv are not sufficient for compensating for the rotation changes. It calls for additional research for representing and matching engineering diagrams that are less sensitive to visual changes and more aware of the semantics of parts in engineering diagrams.

5.6 DISH Summary

A generally applicable neural ranking models should be capable of solving common search issues faced by a wide range of search tasks. This chapter investigates whether the neural ranking model architectures we developed previously can also be used on the engineering diagram search task. Engineering diagram search share some similarities with text search in that the query is matched to a small region of a diagram, making interaction-based neural ranking models a potential solution. It is also very different from text ranking problems because the features are different, the document has a 2D organization instead of 1D, and matching regions can have different scale and rotation which has no analogue for text.

This chapter first presents an unsupervised model, DISH-HP, which is based on the DRMM model (Guo et al., 2016a). It leverages local region features extracted from the middle layer of a deep convolutional neural network instead of word embeddings. It calculates the similarity scores between query regions and document regions, and uses histogram pooling to combine evidence. The local match signals allow small differences to be recognized. Experimental results showed that matching local, region-level signals leads to superior performance over several retrieval algorithms based on global, image-level features.

The chapter then presents a supervised model, DISH-Conv, to improve the robustness to scale and rotation changes. It is based on our Conv-KNRM, which uses convolutional networks to generate n-gram embeddings from word embeddings. DISH-Conv adapts that idea and employs spatial convolutions to generate larger-scale representations from small-scale representations, enabling the query and the document diagrams to match at multiple scales. A novel gating network is proposed to improve the search accuracy by automatically detecting unimportant regions of an image. Experimental results show that DISH-Conv improves the ranking of DISH-HP substantially, especially when the target part appears at a different scale and/or is rotated to a different angle in the document diagram.

Our analysis reveals that the power of DISH-Conv is a combination of its ability to transform features through the convolutions, match at multiple scale levels, and suppress unimportant regions through the gating network. The spatial convolution in DISH-Conv generates multi-scale representations that are able to model changes in scale. The convolution is also able to learn a feature transformation that compensates for moderate changes of rotation. The gating network is able to capture the meaning of a region from its feature vector, and suppresses noisy match signals from unimportant regions. The convolution and the gating network are trained end-to-end so that they automatically learn how to transform features and assign importance for the specific task and dataset.

Traditionally, retrieving different data requires different representations, feature extractors and retrieval models. Designing new systems for every new problem is time-consuming. This chapter identified important similar architectural elements for two very different kinds of data – text and engineering diagrams – demonstrating the generalization ability of the neural ranking architectures developed in this dissertation. It shows that a text neural ranking model can be adapted to images with only small modifications, providing evidence for future research to use neural networks to simplify and unify retrieval systems across data modalities.

Chapter 6

DocBERT Reranker: General Language Understanding for Ranking[1]

Neural ranking models trained on large scale relevance labels have shown promising performance on various ranking benchmarks (Chapter 3-4). One of the key advantages of these methods is the ability to learn search-specific relevance patterns, such as soft match pairs of words and phrases. However, their effectiveness is limited by the accessibility to large scale relevance labels, and does not generalize well to tail queries (Chapter 3) or new search domains (Chapter 4).

The domain adaptation experiment in Chapter 4 shows that some soft match patterns learned by the neural ranking models encode universal properties of human languages and human knowledge, such as ("atypical squamous cells", "cervical cancer"). This type of knowledge can be learned from documents without relying on user queries or clicks. We hypothesize that a more generalizable ranking model should be able to understand the *text content* of queries and documents in addition to the search-specific relevance patterns.

Traditionally, neural ranking models obtain such general-purpose knowledge by loading a pre-trained word embedding, e.g., word2vec (Mikolov et al., 2013). These embeddings are learned from word co-occurrence signals in a corpus, providing hints about synonyms and related words. Nevertheless, word co-occurrence is only a shallow bag-of-words understanding of the text, and is often not sufficient for ranking (Chapter 3). Recently, we have seen rapid progress in text understanding with the introduction of pre-trained neural language models such as ELMo (Peters et al., 2018) and BERT (Lee et al., 2019a). Different from traditional word embeddings, they pre-train a much deeper and larger neural network on the corpus. While the traditional word embeddings mostly only encode word co-occurrence, these deep models can build the context of words, capturing more complex word interactions, dependencies and structures. Fine-tuning these pre-trained deep networks has outperformed traditional neural models on a variety of NLP tasks (Peters et al., 2018; Lee et al., 2019a).

The deep pre-trained language models bring new possibilities to improve IR models' general-purpose language understanding ability, and consequently, their generalization ability. This chapter seeks to understand the performances and properties of BERT (Lee et al., 2019a) in ad hoc ranking tasks, with a focus on low resource scenarios with only limited relevance labels for training.

An open research problem in applying BERT to document ranking is how to handle the input length limitation — the time and space complexity of BERT grow quadratically to the length of the input text, leading to long training time and out-of-memory issues when applied to longer documents [2]. To address

[1] This chapter is based in full on a previously published paper (Dai and Callan, 2019b) appearing in SIGIR 2019.

[2] Due to the high memory complexity, the current implementation of BERT only supports to up to 512 totkens.

this challenge, we present the DocBERT reranker, a passage-based BERT ranking model for document ranking. DocBERT uses a passage ranking framework that uses BERT to score one passage at a time, and combines passage scores to rank document. Document-level summaries, such as document titles, are added to each passage to provide global context. The passage-level BERT model is trained on distant supervision from document-level labels, eliminating the need to label individual passages.

We examine the DocBERT reranker on two ad-hoc retrieval datasets, both with only limited training data. Experiments show that fine-tuning DocBERT models with a small number of relevance labels can achieve better performance than strong baselines, demonstrating that general-purpose language knowledge is critical and can compensate for lacking search-specific training data. The effectiveness of DocBERT improves substantially with domain adaptation, showing that search-specific knowledge is also critical, but they can be learned from related search domains when the target domain lacks relevance labels.

Our further analysis finds that DocBERT excels in *understanding the content of queries and documents* — capturing sentence structures, understanding term dependencies, and adding context to terms using information from other terms in the same a query or a document. This is novel compared to prior neural IR models that focus on query-document interactions. The superior content understanding ability helps DocBERT to tackle several problems that were considered difficult in traditional IR, such as prepositional phrases, long queries, stopwords and punctuation.

The rest of this chapter is organized as the follows. Section 6.1 introduces related work on pre-trained language models and passage-based information retrieval. Section 6.2 introduces the DocBERT reranker. Sections 6.3-6.4 presents experimental methodologies and result. 6.5 concludes this chapter.

6.1 Related Work

This section provides background on pre-trained language models and BERT (Lee et al., 2019a). It also discusses passage-based information retrieval, which is related to the proposed DocBERT reranker that uses a passage ranking framework.

6.1.1 Pre-trained Language Models and BERT

Deep neural networks are prone to overfitting on small training data. Pre-training aims to address this challenge by learning a model on one task to help it form parameters that can be used in other tasks. In this way, the pre-trained knowledge helps the model perform new tasks from old experience instead of from scratch.

Neural language model pre-training has evolved significantly over the past few years. Earlier work on pre-training focused on learning word embeddings from the surrounding context signals in large corpora, such as word2vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). Although these pre-trained embeddings can capture certain linguistic properties of words, they cannot model how a word interacts with its context. Consequently, they often require additional task-specific layers to capture higher-level context, which needs to be trained from scratch on the downstream task. Recently, pre-trained language models have been advanced from shallow to deep. New pre-trained language models such as ELMo (Peters et al., 2018), BERT (Lee et al., 2019a), RoBERTa (Liu et al., 2019) and XLNet (Yang et al., 2019) train very deep neural networks on the corpora. Same as the earlier word embedding approaches, these models are trained on the surrounding context signals, learning general-purpose language patterns in an unsupervised way. Differently, they use much deeper models and therefore capture more complex language structures than the previous word embeddings. These models have brought significant performance gains

on a variety of NLP tasks. One of the most successful and widely-used pre-trained models is BERT (Lee et al., 2019a), as we will discuss next.

BERT is a unified model that can be used across various tasks involving one piece of text (e.g., sentiment classification, named entity recognition) or two pieces of text (e.g., question answering and natural language inference). BERT takes as input a segment of text or a concatenation of two segments. After tokenization, a special token `[CLS]` is added to the beginning of the tokens, which aims to capture features for the entire input sequence. When the input contains two segments, the two segments are concatenated into a single input sequence to BERT with a `[SEP]` token delimiting them.

$$\text{One Segment:}\texttt{input} = \texttt{[CLS]}\ \textit{segment 1}\ \texttt{[SEP]} \tag{6.1}$$

$$\text{Two Segments:}\texttt{input} = \texttt{[CLS]}\ \textit{segment 1}\ \texttt{[SEP]}\ \textit{segment 2}\ \texttt{[SEP]} \tag{6.2}$$

BERT uses the *Transformer* architecture (Vaswani et al., 2017) to model the text sequence. The tokens are first mapped to embeddings, and then processed through several layers of Transformer units, which uses a self-attention mechanism to gradually add context information into tokens' embeddings. The attention takes as input a sequence of embeddings $H = [h_1, ..., h_n]$ corresponding to the n tokens of the input sentence, and transformed each token's embedding h_i into query, key, and value vectors q_i, k_i, v_i through separate linear transformations. It than considers *every pair* of tokens in the input, and computes an attention weight for the pair:

$$\alpha_{ij} = \frac{exp(q_i^T k_j)}{\sum_{l=1}^{n} exp(q_i^T k_l)}.$$

Attention weights are used to generate a new embedding for each token by computing a weighted sum of the all tokens' value vectors:

$$o_i = \sum_{j=1}^{n} \alpha_{ij} v_j.$$

Attention weights can be viewed as governing how "important" every other token is when producing the next representation for the current token. Transformer uses multiple attention heads, i,e, multiple sets of the query, key, and value vectors q_i, k_i, v_i, which can capture different types of relationships between tokens.

BERT is pre-trained on two unsupervised tasks. The first task is the *Masked Language Model* (MLM) task, a cloze task that masks some percentage of the input tokens at random, and then asks BERT to predict those masked tokens. The second task is the *Next Sentence Prediction* (NSP) task. It gives BERT two sentences and asks BERT to predict whether the second sentence follows the first in the original text or is randomly sampled from other text. The Next Sentence Prediction task forces BERT to understand the relationship between two sentences, which is not directly captured by masked language modeling. It make BERT potentially a good fit for *information retrieval*, which also needs to model the relationship between two pieces of text (a query and a document).

At the time of this work, researchers just started investigating BERT for information retrieval. Nogueira and Cho (2019) were one of the first to apply BERT for ranking, demonstrating that fine-tuning BERT and treating ranking as a classification problem outperforms existing neural ranking models by large margins on a passage ranking task. At the same time, BERT's effectiveness on standard document retrieval tasks – where documents are longer and training data are sparse – remains to be studied.

The success of BERT attracted intensive research into the field of pre-training deep language models, and led to many new pre-training models and techniques. For example, RoBERTa (Liu et al., 2019) improves BERT by dynamic masking. XLNet (Yang et al., 2019) proposed a Permuted Language Modeling task that avoids using special masking tokens in pre-training which may cause train-test gap. ALBERT (Lan et al., 2019) uses a sentence order prediction task that is a harder task than the original next

sentence prediction. In terms of efficiency, a family of model compression algorithms are proposed to convert the original BERT into smaller neural networks (Sun et al., 2019b; Jiao et al., 2019). These advances in language model pre-training are concurrent with this thesis research. The rest of this dissertation (Chapters 6-8) mainly uses BERT. We believe the methods we developed are also compatible with these newer models, and they may get further improvements in accuracy and speed by using these enhanced and accelerated models.

6.1.2 Passage-based Information Retrieval

There is rich prior work using passages for document retrieval. The types of passages explored by researchers can be grouped into three classes: discourse, semantic, and window (Callan, 1994). Discourse passages are based upon textual discourse units (e.g. sentences, paragraphs and sections). Semantic passages are based upon the subject or content of the text. Window passages consists of a fixed number of words, which may not take logical structure of the document into account. Prior research shows that overlapping, fixed-size passages of 100-300 words more are effective than semantic or discourse based passages (Callan, 1994; Kaszkiel and Zobel, 1997).

The most widely used way is to *combine passage scores*. This type of approaches estimates the relevance score between queries and individual passages, and aggregates passage scores into a document score. For example, Liu and Croft (2002) choose the highest passage-level relevance score of all passages as the document-level relevance score. Wang and Si (2008) combined the the passage-level scores by also taking into consideration the similarity correlations among the passages. Wu et al. (2019) revisited this topic and designed new rules for combining passage scores by analyzing the manual relevance judgements on passages and documents. Most of the studies discussed here were based on traditional information retrieval techniques, using classic bag-of-words retrieval models to score passages and heuristics to combine passage scores.

Recently, there also emerged several neural ranking models using hierarchical neural networks to capture passage-level relevance information for document ranking, e.g., (Pang et al., 2017b) and (Fan et al., 2018). These models train passage-level relevance models and the document-level ranking model in an end-to-end manner, which learns how to score passages and combine passage scores in a supervised way. However, these hierarchical neural models mostly use small neural networks, and are not compatible with BERT which consumes much larger memory.

6.2 DocBERT reranker

BERT uses the Transformer architecture that considers all possible token pairs in the input. It provides rich signals, but comes at a cost of memory and speed as the computation complexity is quadratic to the input length. As a result, the standard BERT implementation has an input length limitation of 512 tokens, which is about 300 to 400 English words before tokenization. This is shorter than a the typical document length in standard document retrieval tasks. We propose the `DocBERT` reranker, a passage-based ranking framework that adapts BERT for document ranking.

Passage-Level Relevance Modeling. Given a document d, the `DocBERT` reranker first splits d into a sequence of passages:

$$P = \{p_1, ..., p_n\}.$$

Prior research shows that overlapping, fixed-size passages of 100-300 words more are effective than natural passages (Callan, 1994; Kaszkiel and Zobel, 1997). Therefore, `DocBERT` generates passages using a 150-word sliding window with a stride of 75 words.

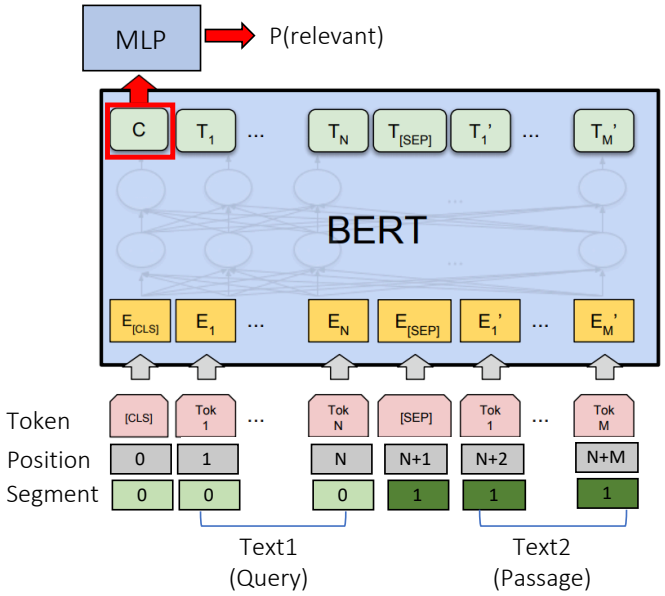


Figure 6.1: `DocBERT` reranker architecture. The figure is based on the original BERT illustration from (Lee et al., 2019a). Query tokens and passage tokens are concatenated. Token embeddings are added with position embeiddings and segment embeddings. The transformers learn interactions among query and document tokens. A multi-layer perceptron (MLP) is used to predict the relevant label.

Next, as illustrated in Figure 6.1, the `DocBERT` reranker uses the BERT two sentence classification architecture to estimate the relevance between a query and each passage. However, the individual passages are likely to lose the global context of the document. For example, a document about "dog" may have a small piece of text discussing "cat" as an comparison to dogs. It will cause the model to mistakenly consider this passage as being relevant to "cat". We would like to add a short summary about the document, providing a global view for BERT. `DocBERT` use document titles as the summary. When title is available, it adds the title to `DocBERT` beginning of every passage to provide global context, and uses two new special tokens `[title]` and `[body]` to indicate the start of the document title and the passage body.

Following Lee et al. (2019a), we join the query and the passage by applying wordpiece tokenization, separating them with [SEP] tokens, prefixing a [CLS] token, and appending a final [SEP] token.

$$\texttt{input} = \texttt{[CLS]}\textit{query}\texttt{[SEP][title]}\textit{title}\texttt{[body]}p_i\texttt{[SEP]} \tag{6.3}$$

The tokens go through several layers of Transformer units. At each layer, a new contextualized embedding is generated for each token by weighted-summing all other tokens' embeddings using the self-attention technique described in Section 6.1. `DocBERT` use the output embedding of the [CLS] token as a representation for the entire query-document pair. It is fed into to a single layer feed-forward neural network to obtain the probability of the passage being relevant.

$$C_{\texttt{[CLS]}} = \texttt{BERT}(\texttt{input})_0 \tag{6.4}$$

$$s(q, p_i) = \sigma(C_{\texttt{[CLS]}}W + b) \tag{6.5}$$

`input` is the input text sequence generated from Eq. 6.3 , σ denotes the sigmoid function, and $s(q, p_i)$ denotes the probability score between a query q and a passage p_i.

To understand the self-attention, we can decouple the attention over the entire sequence into three parts: the attention over the query, the attention over the title and passage, and the attention across the query and the passage. The first part modifies a query token's embedding based on the rest of the query, which can be viewed as *query understanding*. Similarly, the attention over the title and the passage can be viewed as *document understanding*. One novelty here is that the attention spans over the title and the passage, so that it can use the title to better provide context for understanding the passage. Finally, the attention between query tokens and document tokens models the *query-document interaction*. Therefore, the architecture of `DocBERT` reranker is essentially an *interaction-based* neural ranking model, same as K-NRM(3) and Conv-KNRM(3). Differently, `DocBERT` reranker also has the query/document understanding parts that can complex language structures in the query/document, while K-NRM and Conv-KNRM ignores context or only model short-range ones (n-grams).

Passage-Level Evidence Aggregation. `DocBERT` reranker predicts the relevance of each passage independently. To estimate the relevance of the entire document, we experimented with three approaches to aggregating the passage-level evidence.

- `DocBERT-FirstP`$(q, d) = s(q, p_1^d)$: Document score is the score of the first passage.
- `DocBERT-MaxP`$(q, d) = \max s(q, p_i^d)$: Document score is the score of the best passage.
- `DocBERT-SumP`$(q, d) = \sum s(q, p_i^d)$: Document score is the sum of all passage scores.

DocBERT reranker than ranks documents by the aggregated scores. Because DocBERT reranker models the interactions between the query and the passages, it has a relatively high complexity and therefore is used in the reranking stage.

Document-Level Distant Supervision. `DocBERT` reranker is trained on a binary classification task to predict if a query-passage pair is relevant or not. The model is initialized with a pre-trained BERT model[3] to leverage the pre-trained language model, while the last MLP layer is learned from scratch. During training, the entire model is tuned to learn more IR-specific representations.

Training requires relevance annotation for every query-passage pair. However, in many retrieval tasks, we can only collect document-level relevance labels. For example, in a standard web search application, document-level labels can be collect fairly easily from search logs, while it is unclear how to get user feedback on individual pieces of text. This work proposes a *distant supervision using the document-level labels*. The distant supervision considers all passages from a relevant document as relevant and vice versa. We first collect relevant documents (positive examples) and irrelevant documents (negative examples) for each query that appears in the training set. The relevant documents are all the documents labeled as relevant by human annotators. We sample irrelevant documents (negative examples) from the top-ranked documents returned by the first-stage retrieval stage, because `DocBERT` reranker is used to rerank those documents. The sampling follows a uniform distribution over the top 1000 documents in the initial ranking lists excluding the relevant ones, and uses a probability of 10%. To avoid over-fitting, we additionally perform sub-sampling on the passages. For every document in the training set, we use the first passage, the last passage, and 10% of the other passages.

6.3 Experimental Setup

The experiments use two standard text retrieval collections with different characteristics: Robust04 and ClueWeb09-B.

[3]We use uncased BERT-Base model from `https://github.com/google-research/bert`

Table 6.1: Example of Robust04 search topic (Topic 697).

Title	air traffic controller
Description	What are working conditions and pay for U.S. air traffic controllers?
Narrative	Relevant documents tell something about working conditions or pay for American controllers. Documents about foreign controllers or individuals are not relevant.

Robust04 is a news corpus with 0.5M documents and 249 queries. Two versions of queries are included: a short keyword query (*title*) and a longer natural language query (*description*). A *narrative* is also included as the guidance for relevance assessment. An example is shown in Table 6.1.

ClueWeb09-B contains 50M web pages and 200 queries with title and description. Passages are generated using a 150-word sliding window with a stride of 75 words. We evaluate the search effectiveness of ranking models on title queries and description queries; robust04 narratives are used to gain better understanding the model's language understanding ability. Both datasets have limited relevance labels (200-250 queries), making pre-trained models desirable.

The two datasets represent a *low-resource* condition. Previously, we train and test our models using Sogou-Log (Chapter 3) and Bing-Log (Chapter 4) which have hundreds of thousands of unique queries and millions of search sessions. In contrast, Robust04 and ClueWeb09-B only have a few hundreds of unique search queries. This setting allows us to examine if the pre-training of BERT can reduce the need of search-specific annotations.

We compare `DocBERT` reranker to a wide range of baselines, including unsupervised bag-of-words retrieval models, feature-based learning-to-rank models, as well as previous state-of-the-art neural ranking models.

- Unsupervised baselines use Indri's bag of words (`BOW`) (Strohman et al., 2005) and sequential dependency model queries (`SDM`) (Lavrenko and Croft, 2001). These retrieval models were applied to the full document following standard document retrieval practices.
- Feature-based learning-to-rank baselines include `RankSVM` [4] and `Coor-Ascent` [5]. They used the same features as in Section 4.2.
- Neural baselines include `DRMM` (Guo et al., 2016a) and `Conv-KNRM` (Chapter 4). `DRMM` uses the pre-trained word2vec (Mikolov et al., 2013) to model word soft-match. `Conv-KNRM` learns n-gram embeddings for the search task. These two models were shown to be among the best performing neural models on our two datasets (Lin, 2019).

`DocBERT` models are based on the implementation released by Google[6]. Baselines use standard stopword removal and stemming; `DocBERT` uses raw text. Supervised models are used to re-rank the top 100 documents retrieved by `BOW` with 5-fold cross-validation.

Evaluation used NDCG@20 evaluated by the `trec_eval` software, which is the standard TREC evaluation tool for these datasets. It is different from the `gdeval` evaluation software used in Section 3-4. The two softwares apply different normalizations to the NDCG scores. Therefore, the ClueWeb09-B results reported in this chapter are slightly higher than the ones reported previously, but the relative order among the ranking models remains the same. Source code and related resources are released [7].

[4] `https://www.cs.cornell.edu/people/tj/svmlight/svmrank.htm`
[5] `https://sourceforge.net/p/lemur/wiki/RankLib/`
[6] `https://github.com/google-research/bert`
[7] `https://github.com/AdeDZY/SIGIR19-BERT-IR`

Table 6.2: DocBERT's effectiveness on Robust04 and ClueWeb09-B. † indicates statistically significant improvements over Coor-Ascent by permutation test with $p < 0.05$.

	nDCG@20			
	Robust04		ClueWeb09-B	
Model	Title	Description	Title	Description
BOW	0.417	0.409	0.268	0.234
SDM	0.427	0.427	0.279	0.235
RankSVM	0.420	0.435	0.289	0.245
Coor-Ascent	0.427	0.441	**0.295**	0.251
DRMM	0.422	0.412	0.275	0.245
Conv-KNRM	0.416	0.406	0.270	0.242
DocBERT-FirstP	0.444†	0.491†	0.286	**0.272**†
DocBERT-MaxP	**0.469**†	**0.529**†	0.293	0.262†
DocBERT-SumP	0.467†	0.524†	0.289	0.261

6.4 Results and Discussion

Four experiments study the effectiveness of the DocBERT reranker, the source of effectiveness, the impact of adapting DocBERT across search domains, and the difference among different types of queries.

6.4.1 Effectiveness of Pre-trained DocBERT reranker

Table 6.2 reports the accuracy of DocBERT reranker and baseline methods. On Robust04, DocBERT models consistently achieve better search accuracy than the baselines, with a 10% margin on queries and a 20% margin on the longer description queries. On ClueWeb09-B, DocBERT is comparable to Coor-Ascent on title queries, and better on description queries. The results demonstrate the effectiveness of BERT for document retrieval, especially on the longer description queries.

Among the neural rankers, Conv-KNRM has the lowest accuracy. Conv-KNRM needs to learn n-gram embeddings from scratch. Its performance is strong when trained on a large search log, outperforming learning-to-rank baselines RankSM and Coor-Ascent by larger margins (Chapter 4). However, as shown in Table 6.2, its performance is inferior when training data is limited. DocBERT is pre-trained and is less prone to overfitting. DRMM represents words with pre-trained word embeddings. The better performance of BERT models demonstrates that the contextualized text representations are more effective for IR than bag-of-words embeddings.

Comparing the two datasets, DocBERT models perform better on Robust04 than on ClueWeb09-B. This is probably due to that Robust04 is closer to the pre-trained model. Robust04 has well-written articles; its queries look for facts that depend largely on understanding text meaning. ClueWeb09-B documents are webpages that include tables, navigation bars, and other discontinuous text. The task also involves web-search specific issues such as page authority. More training data may be required to learn such search-specific knowledge. We investigate this possibility in Section 6.4.2

Comparing the two types of queries, DocBERT models have larger gains on the longer description queries than on the short title queries. On Robust04, using description queries with DocBERT-MaxP brings a 23% improvement over the best title query baseline (SDM). Most other ranking methods only get similar or worse performance on descriptions compared to titles. To the best of our knowledge, this is the first time we see that description queries outperform title queries with such a large margin. On ClueWeb09-

Table 6.3: Accuracy of Bing augmented DocBERT on ClueWeb09-B. †: statistically significant improvements over Coor-Ascent.

Model	Knowledge		nDCG@20	
	Text	Search	Title	Desc
Coor-Ascent	Low	Low	0.295	0.251
DocBERT-FirstP	High	Low	0.286	0.272†
Conv-KNRM+Bing	Low	High	0.314 †	0.275 †
DocBERT-FirstP+Bing	High	High	**0.333**†	**0.300**†

B, DocBERT manages to shrink the gap between titles and descriptions. Although intuitively, description queries should carry richer information, it is hard to fully utilize them in traditional bag-of-words methods due to difficulties in estimating term importance. Our results show that longer natural language queries are indeed more expressive than keywords, and the richer information can be effectively leveraged to improve search using a deep, contextualized neural language model. Further analysis of DocBERT's ability to understand different types of search queries is given in Section 6.4.3.

In summary, this experiment shows that low-resource search domains can benefit from a pre-trained deep langauge model like BERT. With only a few hundred of training queries, our baseline neural ranking models fail to outperform a feature-based learning-to-rank model, while the DocBERT reranker shows competitive performance and achieve new state-of-the-art performance on Robust04. In addition, we found that the DocBERT reranker performs better on longer, natural language queries, indicating that DocBERT might have a stronger way of modeling, parsing, and analyzing query contents.

6.4.2 Domain Adaptation

The previous experiment demonstrates that for a low-resource search domain without limited training data, fine-tuning a pre-trained BERT can lead to a strong reranker when the domain is similar to pre-training tasks (e.g., Robust04), but the performance is less competitive when the domain requires more search-specific knowledge (e.g., ClueWeb09-B).

This observation is consistent with our previous findings (Chapter 3) that corpus-trained text representations do not always align with the search task. In Chapter 4, we show that Conv-KRNM can learn search-specific patterns from related domains when the target domain has few training data. This experiment investigates if BERT's language modeling knowledge can also be stacked with additional search knowledge learned in a domain-adaptation manner to further alleviate low-resource problems.

This experiment used the methodologies used in the Conv-KRNM domain adaption experiment (Section 4.2): adapting from the Bing-Log to ClueWeb09-B. It first fine-tuned DocBERT, initialized with BERT, on the Bing-Log which contains 0.1 million queries and their relevant documents. It then fine-tuned the DocBERT on ClueWeb09-B using around 150 search queries. We refer to this model as DocBERT-FirstP+Bing.

Table 6.3 shows the domain adaptation results. DocBERT-FirstP is the best in-domain BERT model on ClueWeb09-B (Table 6.2). Its pre-trained language model encodes general word associations like ('Honda', 'car'), but lacks search-specifc knowledge like ('Honda', 'special offer'). Conv-KNRM+Bing was the previous state-of-the-art domain adapted neural IR model on ClueWeb09-B (Chapter 4). It was trained on millions of query-document pairs, but does not explicitly model general language patterns. DocBERT-FirstP+Bing achieves the best performance, demonstrating that BERT is compatible with domain adaption. For search tasks where relevance labels are limited, off-the-shelf pre-trained BERT

Table 6.4: An example showing queries that require different levels of text understanding. The queries are generated from Robust04 search topic 697.

Title	air traffic controller
Description	What are working conditions and pay for U.S. air traffic controllers?
Description, keywords	working conditions pay U.S. air traffic controllers
Narrative	Relevant documents tell something about working conditions or pay for American controllers. Documents about foreign controllers or individuals are not relevant.
Narrative, keywords	relevant documents tell working conditions pay American controllers documents foreign controllers individuals relevant
Narrative, positive	Relevant documents tell something about working conditions or pay for American controllers.

provides general-purpose text understanding knowledge, while the search knowledge can be enhanced by learning from related search tasks in a simple yet effective way.

6.4.3 Understanding Natural Language Queries

The experiment in Section 6.4.1 finds that the DocBERT reranker performs better on longer, natural language queries, which is rarely seen in prior research. The third experiment further examines DocBERT on five types of queries that require different levels of text understanding.

Table 6.4 gives an example of the five types of queries used in this experiment. The first 3 types of queries are the title query that uses a few keywords, the description query that uses a sentence or a question, and the narrative query that uses multiple sentences. To test the effect of grammar structures, a keyword version of description and narrative is generated by removing stopwords and punctuation. To test how DocBERT understands the logic in narratives, a "positive" version of narrative is generated by manually removing negative conditions (e.g. *"Not relevant are documents..."*). Supervised methods use narratives to re-rank title query initial results due to low recall of BOW on narratives, which gives narratives an advantage over the other types of queries.

Table 6.5 shows the performance of SDM, Coor-Ascent and DocBERT-MaxP on Robust04. SDM works best with titles. Coor-Ascent is moderately better with descriptions and narratives. The two methods weight words solely based on term frequencies. For longer queries that mention many concepts, they fail to identify the most important ones, therefore tend to rank off-topic documents at the top. In contrast, DocBERT-MaxP performs better on longer queries. The attention over the query help the model to understand each individual query word in the context of the entire query, helping the model to identify the key concepts and to understand the relationship between different concepts.

Keywords versions perform better than the original query for SDM and Coor-Ascent. Stopwords are noisy to traditional bag-of-words retrieval models that rely on term frequency signals, therefore it is a standard practice to remove stopwords before retrieval. In contrast, DocBERT is more effective on the original natural languages queries that have stopwords and punctuation. Although stopwords and punctuation do not carry information by themselves, they build the structures in a language. DocBERT is able to capture such structures, achieving a deeper query understanding than flat bag-of-words.

Finally, Table 6.5 also shows the limitations of DocBERT. It is unable to leverage evidence from negative logic conditions in narratives. Removing negative conditions does not hurt performance.

Table 6.5: Accuracy of DocBERT on Robust04 using queries of varying length and complexity . Percentages show relative gain/loss over title queries.

Query	Avg Len	nDCG@20					
		SDM		Coor-Ascent		BERT-MaxP	
Title	3	0.427	–	0.427	–	0.469	–
Description	14	0.404	-5%	0.422	-1%	0.529	+13%
Description, keywords	7	0.427	-0%	0.441	+5%	0.503	+7%
Narrative	40	0.278	-35%	0.424	-1%	0.487	+4%
Narrative, keywords	18	0.332	-22%	0.439	+3%	0.471	+0%
Narrative, positive	31	0.272	-36%	0.432	+1%	0.489	+4%

6.4.4 DocBERT Visualization

The last experiment aims to understand how DocBERT reranker models the relevance between a query and a document by visualizing the model. Figure 6.2 visualizes two layers from the DocBERT-MaxP model when predicting the relevance between a description query *"Where are wind power installations located?"* and a sentence *"There were 1,200 wind power installations in Germany"*.

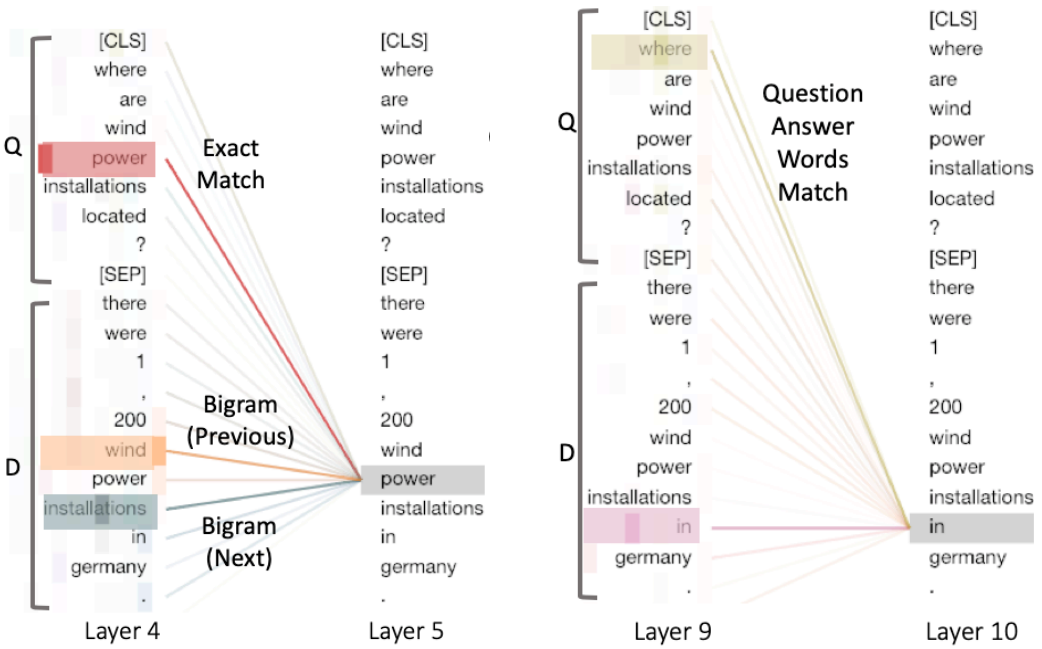
Figure 6.2(a) shows the attention received by the document word *"power"*. The strongest attention comes from *"power"* in the query, which corresponds to exact lexical matching between the query and the document. The token *"power"* also pays strong attention to its previous token and its next token, indicating that the model can build bi-gram representations with the attention. As shown by K-NRM (Chapter 3) and Conv-KNRM (Chapter 4), local matching of words and n-grams are strong neural IR features. This example shows that DocBERT is also able to capture them.

Figure 6.2(b) confirms our findings in Section 6.4.3 that stopwords actually provide important evidence about relevance in DocBERT. It shows that the document token *"in"* receives the strongest attention from the query token *"where"*. The token *"in"* appears in the context of *"in Germany"*, so it satisfies the *"where"* question. This types of matching is difficult in traditional bag-of-words retrieval because the word *"in"* has various uses and does not always indicate answers to a *"where"* question, but a standard bag-of-words retrieval ignores the context around *"in"*. BERT uses self-attention to capture the context of these words, and therefore knows the contribution of each specific word mention.

To summarize, this experiment found that BERT can extract a variety of effective patterns for information retrieval. The attention across the query and the document models the query-document interactions, capturing important interaction patterns such as exact lexical match. The self-attention within the query or the document provides a deep understanding for the *content* of queries and documents, generating several new features for IR, such as evidence from stopwords.

6.5 DocBERT reranker Summary

Most prior neural IR methods focus on learning query-document relevance patterns, and trains the neural models on search-specific labels. However, such approaches require large amounts of training data which are expensive to obtain. Recent deep language models show promising results on learning general-purpose language patterns from large-scale, unsupervised pre-training. This chapter investigates whether the pre-trained deep language models can improve low-resource search domains where training data are limited, and how the pre-trained language knowledge impacts information retrieval.



(a) (b)

Figure 6.2: Visualization of DocBERT reranker. It shows DocBERT's attention scores for a Robust04 description query *"Where are wind power installations located?"* and a relevant document *"There were 1,200 wind power installations in Germany..."*. (a) shows the attention between the 4th layer and the 5th layerof DocBERT. (b) shows the attention between the 9th layer and the 10th layer of DocBERT. Colors represent different attention heads; deeper color indicates higher attention.

Our research is based on BERT, the state-of-the-art pre-trained deep language model at the time. We first propose the DocBERT reranker which adapts BERT for document ranking. It uses a passage-based retrieval framework to addresses the input length limitation of BERT. DocBERT scores passages independently and combines passage scores for document ranking. A distant supervision technique is proposed to train the model using document-level relevance labels, which are often easier to collect than passage-level labels.

We tested DocBERT on two retrieval datasets with distinct characteristics. Our experiments show that the DocBERT reranker achieves new state-of-the-art performance on several evaluation sets. It proves the effectiveness of DocBERT's passage-based framework and distant supervision. DocBERT makes it possible to use BERT to rank longer documents in a simple yet effective way.

The two datasets both only contain hundreds of training queries. In this case, all baseline neural ranking models fail to outperform a feature-based learning-to-rank model due to lack of training data. DocBERT's competitive ranking accuracy demonstrate that the pre-trained, general-purpose language knowledge are indeed beneficial to low-resource domains. Further experiment shows that a simple domain adaptation technique can further boost the accuracy of DocBERT on low resource datasets. We believe these results will be a useful reference for IR researcher and developers working on cold-start scenarios and low resource domains.

The DocBERT reranker brings large improvements to natural language queries. People have been trained to use short keyword queries because classic bag-of-words retrieval models cannot effectively extract key information from natural language. Even for systems that do take natural language queries, such

as retrieval-based question-answering systems, a common practice is to convert the query into keywords by filtering out stopwords and punctuations in order to reduce noise. This work found that BERT can capture deeper language structures under the surface form of queries and documents, generating several new features that were difficult to model in traditional IR, such as stopwords and punctuation.

Importantly, `DocBERT` indicates that neural models have two distinct contributions to more accurate retrieval: text understanding, and matching. The first part of this dissertation has focused on *matching*, using neural networks as a black box to model the interactions between queries and documents. In the next part of this dissertation, we will exploit pre-trained language models for *text understanding*, explicitly generating text analysis signals from the query or the document that can be used by downstream IR tasks.

Part III

Query/Document Representation: Context-Aware Term Importance Weighting

Chapter 7

DeepCT: Context-Aware Sentence/Passage Term Importance Estimation[1]

State-of-the-art search engines adopt a pipelined retrieval system: an efficient first-stage that uses a query to fetch a set of documents from the entire document collection, and subsequently one or more reranking stages that refine ranking within the retrieved set. In the past few years, the reranking stage has been improved significantly with the advances in neural ranking models, such as our previous research (`K-NRM`, `Conv-KNRM`, and the `DocBERT` reranker) as well as many efforts from the research field. Meanwhile, these models' power comes from soft matching words in an embedding space, which enables a query to match *every* document, making them cost-prohibitive to be used for initial retrieval that may face billions of documents.

With recent deep language models like BERT pushing reranking accuracy to new levels, the initial retrieval stage is gradually becoming the weak link in modern search engines. We believe it is time to extend neural IR research to the initial retrieval stage, asking the central question of this chapter: *Is it possible to bring deep language understanding into the initial retrieval stage and retain efficiency at the same time?*

Typically, the initial retrieval stage uses a Boolean, probabilistic, or vector space bag-of-words retrieval model that fetches information from an inverted index. To quantify the contribution of each query or document term, most retrieval methods use frequency-based signals such as document and query term frequency (*tf*, *qtf*) to determine the context-specific importance of a term, and inverse document frequency (*idf*) or a similar value to determine its importance globally. Frequency-based bag-of-words retrieval models such as BM25 (Robertson and Walker, 1994) have remained state-of-the-art for decades, and are still the most widely used first-stage retrieval algorithms today.

Though being a huge success, frequency-based term weights are a crude tool. Term frequency does not necessarily indicate whether a term is important or central to the meaning of the text. Table 7.1 shows two passages from the MS MARCO machine reading comprehension dataset (Nguyen et al., 2016). If a user searches for "stomach", the two passages will be considered similarly relevant by frequency-based retrieval models because both mention the query word "stomach" twice; but the second passage is actually off-topic. To identify which terms are central requires a deeper understanding that considers the meaning of a word, the meaning of the entire text, and the role the word plays in the text.

Our previous research on the `DocBERT` Reranker (Chapter 6) gives hints for using pre-trained deep language models as a solution. Compared to traditional bag-of-words approaches, `DocBERT` makes large

[1]This chapter is based in full on a preprint (Dai and Callan, 2019a) and a conference paper (Dai and Callan, 2020a) appearing in SIGIR 2020.

Table 7.1: An example where term-frequency based weighting fails. The two passages each mentions '**stomach**' twice. Only the first passage is about the topic 'stomach'. This chapter proposes a method to weight terms by their roles in a specific text context, as shown by the heatmap over terms. "Hotter" terms are considered more central to the pssage.

In some cases, an upset **stomach** is the result of an allergic reaction to a certain type of food. It also may be caused by an irritation. Sometimes this happens from consuming too much alcohol or caffeine. Eating too many fatty foods or too much food in general may also cause an upset **stomach**.
All parts of the body (muscles , brain, heart, and liver) need energy to work. This energy comes from the food we eat. Our bodies digest the food we eat by mixing it with fluids(acids and enzymes) in the **stomach**. When the **stomach** digests food, the carbohydrate (sugars and starches) in the food breaks down into another type of sugar, called glucose.

improvement on long queries of sentence or passage-length. Analysis reveals that DocBERT leverages word meanings and sentence structures to adjust the attention between words, which can be viewed as a kind of implicit term weighting. Nevertheless, DocBERT is trained and used to directly estimate query-document relevance score, and must be done online in the reranking stage.

This work takes a step further – it seeks to explicitly leverage BERT to analyze the text, estimate term weights, and use them efficiently for initial retrieval. We present the Deep Contextualized Term Weighting framework (DeepCT). DeepCT is trained in a supervised manner to learn a BERT-based contextualized word representation model, as well as a mapping function from representations to term weights. As a word's representation depends on its specific context, the estimated weight for the same term varies with the context. Supporting up to 512 text tokens [2], DeepCT can make use of the linguistic context in sentence/passage-long text to generate more representative term weights, helping the cases where the original term frequency distribution is flat.

One use of DeepCT is to identify *essential terms in passages*. As shown in Table 7.1, passages usually have flat term weight distribution, making term-frequency based retrieval less effective. We develop a novel DeepCT-Index that weights and indexes terms in passage-long documents. The inference step is query-independent, allowing it to be done offline during indexing. The context-based passage term weights are scaled to *tf*-like integers that are stored in an ordinary inverted index that can be searched efficiently by common initial retrieval models.

Another use of DeepCT is to identify *important terms in queries*. Long queries are challenging even in today's commercial search engines. Our DocBERT Re-ranker shows that longer queries actually have good retrieval accuracy in the re-ranking stage, but they are still bottle-necked by the early stage retrieval, where bag-of-words failed to identify the central terms from the many terms and concepts in the query. We follow a query term weighting framework proposed by Zheng and Callan (2015) to develop DeepCT-Query. The predictions are used to generate weighted queries that can be used with widely-used retrieval models.

Our experiments demonstrate that DeepCT generates effective term weights for both documents and queries that lead to large improvements in the initial retrieval stage. Analysis shows that DeepCT's main advantage is its ability to estimate term importance using the meaning of the context rather than term frequency signals, allowing the retrieval model to differentiate between key terms and other frequently

[2]This is limited by the input length limitation of BERT (Lee et al., 2019a).

mentioned but non-central terms. Being a strong initial retrieval model, DeepCT have been adopted by several retrieval systems and helped these systems to achieve state-of-the-art end-to-end accuracy[3]. More importantly, DeepCT shows that term frequency signals – which have been used for 50-60 years in IR – is no longer sufficient, making an important move from "frequencies" to "meanings".

Section 7.1 discusses related work. Section 7.2 describes the Deep Contextualized Term Weighting framework (DeepCT), its use for passage indexing (DeepCT-Index), and its use for for query weighting (DeepCT-Query). Section 7.4-7.5 describe our methodologies and experiments. Section 7.6 concludes.

7.1 Related Work

The initial retrieval stage in modern search engines rely on bag-of-words retrieval models for their efficiency and effectiveness. The earlier retrieval systems used unweighted Boolean match. In 1972, Jones (1972) convincingly demonstrated that the weighting of query terms can significantly improve retrieval performance compared to unweighted match ranking. Ever since, term weighting has been playing a key role in information retrieval research.

Croft and Harper (1979) modeled the query term weight as a tuned constant (the Croft/Harper Combination Match model). Greiff (1998) tried to predict term weight with a linear function of *idf*. Robertson and Walker (1994) proposed the 2-Poisson model based on the assumption that the distribution of within-document frequencies is Poisson for the relevant documents, and also (but with a different mean) for the non-relevant documents. They developed a family of weighting functions that integrate within-document term frequency, within-query term frequency, document length, and collection-wise term frequency. One of the most well-known models is the BM25 retrieval model. Zhai and Lafferty (2004) proposed to apply Jelinek-Mercer smoothing and Dirichlet smoothing methods for statistical retrieval language models. These smoothing methods are effectively equivalent to modeling global term weights from the entire collection. BM25 and query language model with smoothing are two strong retrieval models that are being widely used in today's information retrieval systems.

As can be seen, in standard retrieval models like BM25 and query language model, term weighting relies on frequency-based signals such as term frequency in query (*qtf*), term frequency in document (*tf*), and inverse document frequency (*idf*). Conceptually, they are a simple estimation of how important the term is to the document (document-specific weight), how important the term is to the query (query-specific weight), and how discriminative this term is in general (global term weight). Although effective, frequency signals ignore important evidence such as the context around a word and the meaning of a word. There is rich research developing alternative ways to model term weights, as will be discussed below.

For document-specific term weighting, the most widely-used alternatives to *tf* are graph-based methods (Mihalcea and Tarau, 2004; Blanco and Lioma, 2012; Rousseau and Vazirgiannis, 2013). Graph-based approaches build a document graph, where the nodes represent terms and edges represent term co-occurrence within a maximum distance. Terms are scored by graph ranking algorithms such as PageRank, and scores are used for retrieval (Blanco and Lioma, 2012; Rousseau and Vazirgiannis, 2013). In this way, a word's context can be taken into consideration with the graph. Another alternative way is to *learn* the document-specific weights using regression strategies. Fuhr and Buckley (1991) learned to predict indexing weights from relevance information to provide a better probabilistic index. Simple collection statistics are used as prediction features, for example, *tf*, *idf*, document length, etc. Berkeley Regression (Cooper et al., 1992) learned a whole retrieval model from relevance judgments, where the logistic regression model was also built on top of standard features used by traditional retrieval models such as *tf*

[3]At the time this draft was written, 4 of the top 10 systems on the MS MARCO passage ranking dataset (https://microsoft.github.io/msmarco/) used DeepCT for initial retrieval.

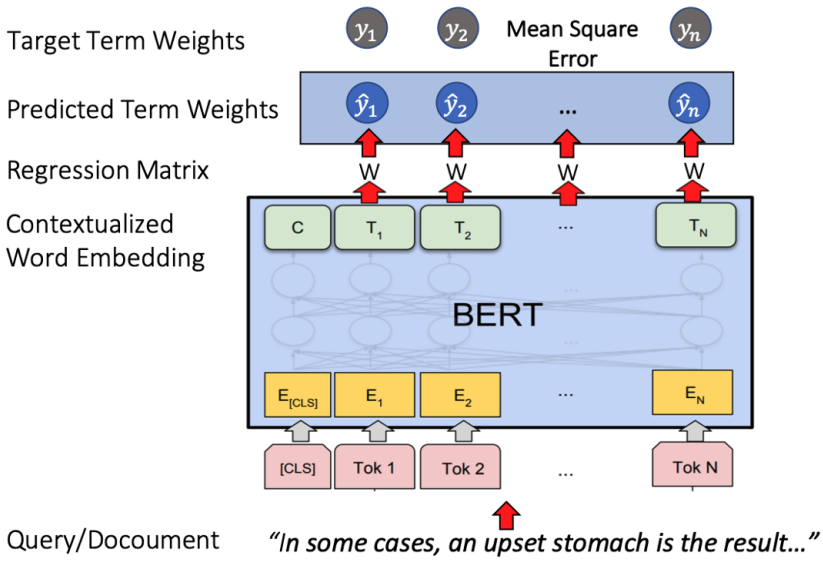


Figure 7.1: The DeepCT framework.

and *idf*. Because of the restricted features, these approaches only show marginal improvements over *tf***idf* baselines, and cannot outperform modern feature-based learning-to-rank methods in most cases.

For query-specific term weighting, a successful approach is to use pseudo-relevance feedback techniques to collect features (Bendersky et al., 2010, 2011, 2012). These methods improve search accuracy compared to frequency-based query term weighting, but the use of pseudo-relevant feedback causes extra computational cost. To predict query term weights from just the text content, Zheng and Callan (2015) proposed a word-embedding based method called `DeepTR`. `DeepTR` constructs a feature vector for each query term using the difference between the term's word2vec (Mikolov et al., 2013) embedding and the average query embedding. It then learns a regression model to map the feature vector onto the term's ground truth weight (term recall weight (Zhao and Callan, 2010)). The estimated weights are used to generate bag-of-words queries that can be searched in the initial retrieval stage.

Recently, a few neural IR models estimate term weights from word embeddings (Dehghani et al., 2017; Guo et al., 2016a), because word embeddings may encode certain aspects of word meanings that are useful for determining word importance. Most of these methods use the embedding-learned term weights in the reranking stage, learning the term weighting function together with the rest of the neural reranker. They also only learn a global *idf* term weight, as the word embeddings do not change with context. It is not clear whether this type of approach can improve the initial ranking stage.

7.2 DeepCT

This section presents the Deep Contextualized Term Weighting framework (`DeepCT`), how it is used to index documents (`DeepCT-Index`), and how it is used to weight query terms (`DeepCT-Query`).

7.2.1 DeepCT Framework

Figure 7.1 shows the DeepCT framework. To estimate the importance of a word in a specific text, the most critical problem is to generate features that characterize a word's relationships to the text context. DeepCT uses BERT for this purpose.

As discussed in Chapter 6.1, BERT uses an attention mechanism where a word gradually absorbs context information based on its attention to every other word in the same text. The resulting contextualized word embedding can be viewed as a feature vector that characterizes the word's syntactic and semantic role in a given context. DeepCT linearly combines the features into a word importance score:

$$\hat{y}_{t,c} = \vec{w} T_{t,c} + b \tag{7.1}$$

where $T_{t,c}$ is token t's contextualized embedding in text c; and, $\vec{w}$ and b are the linear combination weights and bias. When a term is mentioned multiple times in the text, we use its maximum weight. As can be seen, DeepCT is essentially a new type of bag-of-words representation for the text. It shares the advantages of classic bag-of-words document representations: efficient retrieval, support for fine-grained term signals (Guo et al., 2016a), and higher interpretability compared to latent topic models and embeddings. On the other hand, unlike traditional *tf* bag-of-words, DeepCT does not assume term independence when estimating term weights. Building upon BERT's transformer architecture (Lee et al., 2019a), DeepCT takes into account word order, dependencies, and complex interactions.

DeepCT is trained with a per-token regression task, as illustrated in Figure 7.1. Assume that we were given the ground truth term weights in text c – the target term weights that a perfectly-trained DeepCT should produce – denoted as $\{y_{1,c}, \ldots, y_{N,c}\}$. DeepCT tries to minimize the mean square error (MSE) between the predicted weights $\hat{y}$ and the target weights y:

$$loss_{MSE} = \sum_{c} \sum_{t} (y_{t,c} - \hat{y}_{t,c})^2. \tag{7.2}$$

Stopwords, punctuation, and subwords require special handling in DeepCT. Stopwords and punctuation are often discarded by bag-of-words retrieval models. However, Chapter 6 shows that these tokens play an important role in BERT because they are integral to language structure. Therefore, we leave stopwords and punctuation in BERT's input, but set their target weights as 0 to require that the model learns that they are not important to relevance. Subwords are generated from the tokenization step. We use the weight for the first subword as the weight of the entire word; other subwords are masked out when computing the loss.

The entire DeepCT model, from BERT to the regression layer, is optimized end-to-end. DeepCT can *learn* different definitions of term importance depending on how ground truth term weights are defined. The predicted term weights can also be *used* differently depending on the task. Below, we describe two approaches to using the DeepCT framework to improve initial retrieval.

7.2.2 Index Passages with DeepCT

A novel use of DeepCT is to identify terms that are central to the meaning of a document. As shown in the "stomach" example in Table 7.1, classic term frequency signals cannot tell whether the text is centered around a term or just mentions that term when discussing some other topic. This issue is especially difficult in the initial retrieval, where complex features and models are too expensive to apply. We propose DeepCT-Index, a method that uses DeepCT to weight document terms and stores the weights in a typical inverted index. As BERT has a maximum input length limitation of 512 tokens, here we focus on *passages* that can fit into this limitation. Passages and short documents are also the case where weighting

by term frequency is less effective, because the *tf* distribution tends to be flat. Chapter 8 extends this method to longer documents.

Training `DeepCT` *for passage term weighting*. Proper target term weights should reflect whether a term is essential to the passage or not. We propose *query term recall* as an estimation of the ground truth passage term importance:

$$y_{t,c} = QTR(t,d) = \frac{|Q_{d,t}|}{|Q_d|} \in [0,1]. \tag{7.3}$$

Q_d is the set of queries for which passage d is judged relevant, and $Q_{d,t}$ is the subset of Q_d that contains term t. $QTR(t,d)$ is the percentage of queries containing term t. The intuition behind query term recall is that words that appear in relevant queries are more important than other words in the document.

As can be seen Eq. 7.3, to train `DeepCT` requires relevant query-passage pairs. The model takes the text content of a passage, make predictions, and computes the loss based on the relevant query-passage pairs. But once `DeepCT` learns model parameters, it can make estimates for any passage without the need of queries. This allows estimated term weights to be calculated and stored during offline indexing.

Indexing with `DeepCT` *term weights*. We apply the trained `DeepCT` model to all passages in the collection. The predicted weights, which usually falls into the [0..1] range, are scaled into to an integer range that can be used with existing retrieval models. We call this weight tf_{DeepCT} to convey that it is an alternate way of representing the importance of term t in passage d:

$$tf_{\text{DeepCT}}(t,d) = round(\hat{y}_{t,d} * N), \tag{7.4}$$

where $\hat{y}_{t,d}$ is the predicted term weights for term t in passage d. N scales the predicted weights into a integer range. This work uses $N = 100$ as two digits of precision is sufficient for this task.

tf_{DeepCT} is used to *replace* the original *tf* value in the inverted index. The new index, `DeepCT-Index`, can be searched by mainstream bag-of-words retrieval model like BM25 or query likelihood model (QL). The context-based term weight tf_{DeepCT} is expect to bias the retrieval models to central terms in the passage, preventing off-topic passages being retrieved.

In terms of efficiency, the main difference between `DeepCT-Index` and a typical inverted index is that the document-specific term weight uses tf_{DeepCT} instead of *tf*. This calculation is done offline. No new posting lists are created, thus the index does not become larger. To the contrary, a side-effect of `DeepCT` is that tf_{DeepCT} of some terms becomes 0. This can be viewed as a form of static index pruning (Soffer et al., 2001). Research done with colleagues at RMIT shows that the pruning affects of `DeepCT` can actually improve efficiency (Mackenzie et al., 2020), however that work lies outside the scope of the dissertation.

7.2.3 Query Term Weighting with DeepCT

Another straightforward use of `DeepCT` is to weight query terms in long queries. For long queries that mention many terms and concepts, it is important to identify which are central. For example, given the query *"Find locations of volcanic activity which occurred within the present day boundaries of the U.S and its territories"*, an ideal system would understand that *"volcanic activity"* is the key concept, and that *"boundaries of the U.S"* maybe not be necessary in some corpora. Zheng and Callan (2015) proposed a word2vec-based query term re-weighting framework, called `DeepTR`, that efficiently re-weights bag-of-words queries. We follow the `DeepTR` framework, but replace the word2vec-based model with `DeepCT`. We call the proposed approach `DeepCT-Query`.

Training `DeepCT` *for query term weighting*. Following Zheng and Callan (2015), `DeepCT-Query` uses *term recall*l (Zhao and Callan, 2010) as the target term weights to train the query term weighting

model:

$$y_{t,c} = TR(t,q) = \frac{|D_{q,t}|}{|D_q|} \in [0,1]. \tag{7.5}$$

D_q is the set of documents that are relevant to the query. $D_{q,t}$ is the subset of relevant documents that contains term t. Their ratio, $TR(t,q)$, is the percentage of relevant documents that mention term t. Training `DeepCT-Query` with term recall is based on the assumption that a query term is more important if it is mentioned by more relevant documents.

Similar to `DeepCT-Index`, to train `DeepCT-Query` with term recall weights requires relevant query-document pairs. The model takes the text content of a query, makes predictions, and computes the loss with target weights generated from the relevant documents. During inference, the model only needs the query.

Formulating queries with `DeepCT` term weights. When a query is received, the trained `DeepCT` model is used to predict importance weights for each term. Following Zheng and Callan (2015), we use the estimated query term weights to generate bag-of-words queries (BOW) and sequential dependency model queries (SDM) using the Indri query language (Strohman et al., 2005). For example, given the query "CMU campus", the original BOW query in the Indri query language is:

```
#and(cmu campus).
```

Assume the term weights estimated by `DeepCT` are "cmu:0.8" and "campus: 0.3", we reformulate the query as the follows:

```
#weight(0.8 cmu 0.3 campus)
```
[4]

The sequential dependency model (SDM) adds bigrams and word co-occurrences within a window to the BOW query. The orignal SDM query for "CMU campus" is:

```
#weight(0.8 #and(cmu campus)
        0.1 #near1(cmu campus)
        0.1 #window8(cmu campus)),
```

where `#1(cmu campus)` matches when "cmu" and "campus" form a bigram, and `#uw8(cmu campus)` matches when "cmu" and "campus" occur in any order within a window of 8 words. `DeepCT-Query` uses the re-weighted `BOW` query to replace the BOW part of the `SDM` query, as shown below:

```
#weight(0.8 #weight(0.8 cmu 0.3 campus)
        0.1 #near1(cmu campus)
        0.1 #window8(cmu campus)),
```

In terms of efficiency, predicting term weights for a new query is simply feeding-forward the query string through `DeepCT`. We use `Bow-DeepCT-Query` and `SDM-DeepCT-Query` to denote the re-weighted bag-of-words and sequential dependency queries.

7.3 Experimental Methodology for DeepCT-Index

This section presents the experimental methodology for the first task – passage term weighting and indexing.

[4] `#weight` is Indri's probabilistic weighted-AND operator.

7.3.1 Dataset

When we started this dissertation research in 2016, there were no publicly available IR datasets that provide large-scale relevance labels for training the neural networks. Therefore, we created the Sogou-Log and Bing-Log datasets to facilitate our research in Chapter 3- 6, but these search logs cannot be released to the public due to privacy constraints. Later, as neural IR research started to demonstrate the importance of having training data, there were gradually more efforts on building public datasets with large amounts of relevance labels. This work switches to two such passage retrieval benchmarks: the MS MARCO Passage Ranking dataset (Nguyen et al., 2016) and the TREC-CAR dataset (Dietz et al., 2017).

The MS MARCO Passage Ranking dataset (MS-MARCO-Pas) (Nguyen et al., 2016) is a question-to-passage retrieval dataset with 8.8M passages. Average passage length is around 55 words. The training set contains approximately 0.5M pairs of queries and relevant passages. On average each query has one relevant passage. The development (dev) set contains 6,980 queries and their relevance labels. The test set contains 6,900 queries, but the relevance labels are hidden by Microsoft. Therefore, *the dev set is our main evaluation set.* In a few experiments, we also evaluated on the test set by submitting our rankings to the MS MARCO Passage Ranking leaderboard [5].

TREC-CAR (Dietz et al., 2017) consists of 29.7M English Wikipedia passages with an average length of 61 words. Queries and relevant passages are generated synthetically. A query is the concatenation of a Wikipedia article title with the title of one of its sections. Following prior work (Nogueira and Cho, 2019; Nogueira et al., 2019), we use the automatic relevance judgments, which treats paragraphs within the section as relevant to the query. The training set and validation set have 3.3M query-passage pairs and 0.8M pairs respectively. The test query set contains 1,860 queries with an average of 2.5 relevant paragraphs per query.

7.3.2 Baselines and Experimental Methods

We used three baseline indexing methods, as described below.

- *tf* index is a standard *tf*-based index, e.g., as used by BM25.
- `TextRank` (Mihalcea and Tarau, 2004) is a widely-used unsupervised graph-based term weighting approach. We use the open source PyTextRank implementation[6]. Term weights from TextRank are in the range (0, 1); we scale them to integers as described in Eq. 7.4 for indexing.
- `Doc2Query` (Nogueira et al., 2019) is a supervised neural baseline. It trains a neural sequence-to-sequence model to generate potential queries from passages, and indexes the queries as document expansion terms. `Doc2Query` implicitly re-weights terms because important passage terms are likely to appear in the generated queries. We use the `Doc2Query` index for MS-MARCO-Pas released by the authors. No such index is available for TREC-CAR, so we use published values for that dataset (Nogueira et al., 2019).

The baselines were compared to three experimental indexing methods include the proposed `DeepCT-Index` and two variants using different embeddings.

- $\texttt{DeepCT}_{\texttt{W}}\texttt{-Index}$ replaces the BERT component in `DeepCT` with context-independent word embeddings. To provide context to each word, we modeled the passage using the average word embeddings and subtracted the passage embedding from each word's embedding, which was inspired by (Zheng and Callan, 2015). The embeddings are initialized by word2vec (Mikolov et al., 2013) and fine-tuned during training.

[5] The leaderboard is at `https://microsoft.github.io/msmarco/`

[6] `https://github.com/DerwenAI/pytextrank`

- DeepCT$_E$-Index replaces the BERT component in DeepCT with ELMo (Peters et al., 2018). ELMo is a pre-trained, context-aware text representation model that differs from BERT in the network architecture and pre-training task. ELMo was initialized with the pre-trained model described by Peters et al. (2018) and fine-tuned during training.

The BERT part of DeepCT-Index was initialized with pre-trained parameters. For MS MARCO, we used the official pre-trained BERT (uncased, base model) (Lee et al., 2019a). TREC-CAR cannot use the official model because its testing documents overlapped with BERT's pre-training documents. We used a BERT model from Nogueira and Cho (2019) where the overlapping documents are removed from the pre-training data. After initialization, DeepCT is trained for 3 epochs on the training split of our datasets, using a learning rate of $2e^{-5}$ and a max input text length of 128 tokens.

The term weights are stored into a typical inverted index and can be retrieved by any bag-of-words retrieval models. We tested two popular retrieval models: BM25 and query likelihood with Jelinek-Mercer smoothing (QL). We used the Anserini toolkit implementations. We *fine-tuned* BM25 parameters $k_1 \in \{0.1, ...0.9, 1, 2, ..., 12\}$ and $b \in \{0.1, ...0.9\}$, and QL smoothing factor $\lambda \in \{0.1, ...0.9\}$ through a parameter sweep on 500 queries from the training set for *all baselines*. When BM25 was used, the best hyperparamters for the *tf* baseline were $k_1 = 0.6$ and $b = 0.8$, while those for DeepCT-Index were $k_1 = 10$ and $b = 0.9$, indicating that DeepCT-Index needs a higher k_1 and relatively strong document length normalization (controlled by b). When QL was used, the best hyperparamter for *tf* was $\lambda = 0.9$, while for DeepCT it was $\lambda = 0.6$, indicating that the Jelinek-Mercer smoothing is still necessary in DeepCT-Index. These results suggest that more accurate document-specific term importance estimation (better *tf*) cannot simply take the place of the other elements of in the older retrieval models. Concepts like smoothing, length normalization, and global term importance (*idf*) are still important and worth investigation in future neural IR models.

In some experiments, we study the impact of DeepCT's initial document rankings on downstream rerankers. Reranking was done by two rerankers: Conv-KNRM (Dai et al., 2018) and a BERT-based passage reranker developed by Nogueira and Cho (2019). We applied both rerankers to rerank up to 1,000 passages from the initial ranking.

Retrieval and reranking results were evaluated by Mean Reciprocal Rank at 10 passages (MRR@10), the official MS MARCO evaluation metric. For TREC-CAR, we also report MAP at depth 1,000 following the evaluation methodology used in previous work (Nogueira and Cho, 2019; Nogueira et al., 2019). Source code and related data are released [7].

7.4 DeepCT-Index Results

Three experiments investigated the initial retrieval accuracy of DeepCT-Index indexes, its impact on downstream rerankers, and why DeepCT-Index term weights are effective.

7.4.1 Retrieval Accuracy of DeepCT-Index

The first experiment examines whether DeepCT-Index improves initial retrieval accuracy over baseline bag-of-words retrieval models. It also compares the retrieval accuracy of DeepCT-Index to several supervised ranking systems.

Table 7.2 shows the retrieval accuracy of BM25 and QL using indexes generated by six term weighting methods. As can be seen, term frequency (*tf*) is a strong baseline for initial retrieval. The graph-based

[7] https://github.com/AdeDZY/DeepCT

Table 7.2: Ranking accuracy of BM25 and QL using indexes built with three baselines and three DeepCT-Index methods. Win/Tie/Loss are the number of queries improved, unchanged, or hurt, compared to *tf* index on MRR@10. * and † indicates statistically significant improvements over *tf* index and Doc2Query. Doc2Query results for TREC-CAR are from Nogueira et al. (2019); statistical significance for these results is unknown.

Index	MS-MARCO-Pas, dev set			
	BM25		QL	
	MRR@10	W/T/L	MRR@10	W/T/L
tf index	0.191	-/-/-	0.189	-/-/-
TextRank	0.130	662/4556/1762	0.134	702/4551/1727
Doc2Query	0.221*	1523/4431/1026	0.224*	1603/4420/957
DeepCT-Index	**0.243***†	2022/3861/1097	0.230*	1843/4027/1110
DeepCT_{W}-Index	0.174	931/4804/1245	0.168	867/4793/1320
DeepCT_{E}-Index	0.234*†	1891/4139/950	0.220*	1726/4210/1044

Index	TREC-CAR					
	BM25			QL		
	MRR@10	MAP	W/T/L	MRR@10	MAP	W/T/L
tf index	0.233	0.174	-/-/-	0.211	0.162	-/-/-
TextRank	0.160	0.120	167/1252/441	0.157	0.118	166/1327/367
Doc2Query	-	0.178	-/-/-	-	-	-/-/-
DeepCT-Index	**0.332***	**0.246***	615/1035/210	**0.330***	**0.247***	645/1071/144
DeepCT_{W}-Index	0.205	0.147	250/1311/309	0.192	0.139	245/1345/280
DeepCT_{E}-Index	0.280*	0.201*	516/1144/200	0.276*	0.197*	540/1190/130

approach TextRank failed to beat *tf*. Doc2Query – a sequence-to-sequence neural network approach – was effective for MS-MARCO-Pas, but was only marginally better than *tf* on TREC-CAR.

DeepCT-Index outperformed the baselines by large margins. It improved BM25 by 27% on MS-MARCO-Pas and 46% on TREC-CAR. It produced similar gains for QL, showing that DeepCT-Index is useful to different retrieval models. Win/Loss analysis shows that DeepCT-Index improved 25-35% of the queries and hurt 8-16%. To the best of our knowledge, this is the first time an initial retrieval model can get rid of *tf*, use an alternative term weighting method, and generate substantially better rankings.

Results for DeepCT_{W}-Index and DeepCT_{E}-Index demonstrate the importance of context. Non-contextual word2vec embeddings produced term weights that were less effective than *tf*. ELMo produced more effective term weights, but BERT's stronger use of context produced the most effective weights. These results also show the generality of the DeepCT framework, and suggest that it might also be useful with more advanced contextualized text representations such as RoBERTa (Liu et al., 2019) and XL-Net (Yang et al., 2019).

We further compared the initial retrieval results from DeepCT-Index to several supervised ranking systems by participating in the MS MARCO passage ranking competition. Table 7.3 shows results from the MS MARCO leaderboard[9]. It first lists 2 important initial retrieval models: the official standard BM25

[8]The results for Feature-based LeToR are provided by the MS MARCO Passage Ranking task organizers as an official baseline. Results were from the MS MARCO website.

[9]https://microsoft.github.io/msmarco/. May 18, 2020.

Table 7.3: Retrieval accuracy of BM25 with `DeepCT-Index` compared with several representative ranking/re-ranking systems on the MS-MARCO-Pas dataset using official evaluation on dev set and hidden test set. Statistical significance is unknown because the MS MARCO website publishes only summary results.

	Ranking Method	dev MRR@10		test MRR@10	
Single-Stage	Official BM25	0.167	-30%	0.165	-31%
	Doc2Query (Nogueira et al., 2019)	0.215	-12%	0.218	-9%
	DeepCT-Index	0.243	–	0.239	–
Multi-Stage	Official Feature-based LeToR [8]	0.195	-20%	0.191	-20%
	K-NRM (Xiong et al., 2017b)	0.218	-10%	0.198	-17%
	Duet V2 (Mitra and Craswell, 2019)	0.243	+0%	0.245	+2%
	Conv-KNRM (Dai et al., 2018)	0.247	+2%	0.247	+3%
	FastText+Conv-KNRM (Hofstätter et al., 2019)	0.277	+14%	0.290	+21%
	BERT reranker (Nogueira and Cho, 2019)	0.365	+50%	0.359	+50%

and `Doc2Query BM25` (Nogueira et al., 2019). `DeepCT-Index BM25` outperformed both. Table 7.3 also lists representative supervised approaches for feature-based learning-to-rank, previous state-of-the-art neural rerankers (*non-ensemble version*), and BERT-based rerankers. FastText+Conv-KNRM (Hofstätter et al., 2019) was the best non-BERT model on the leaderboard. BERT reranker (Nogueira and Cho, 2019) uses BERT as a black-box model that takes a query-passage pair and outputs a relevance scores; most recently-proposed rerankers in the BERT family (Nogueira and Cho, 2019; Dai and Callan, 2019b) are based on this approach.

A BM25 retrieval from `DeepCT-Index` was better than several systems that used supervised rerankers. It is more accurate than feature-based learning-to-rank, a widely used re-ranking approach in modern search engines. It is also more accurate than a popular neural re-ranking model `K-NRM` (Xiong et al., 2017b). Compared to `Duet V2` (the official re-ranking baseline) and `Conv-KNRM` (Dai et al., 2018), `DeepCT-Index BM25` achieves similar accuracy while being more efficient as it does not need the re-ranking stage. During the past few years, reranking models have been becoming increasingly complex in order to improve accuracy. This work shows that complex rerankers are not the only choice – `DeepCT` takes a different path that moves the complex document understanding process to the offline time, building deep yet simple text representations that can be retrieved very efficiently.

Finally, strong neural rerankers like the BERT-based ones were much more effective than `DeepCT`. These models generate high quality soft-match signals between query and passage words (e.g. "hotel" to "motel"). In contrast, `DeepCT-Index` uses bag-of-words retrieval, which only matches terms exactly and therefore provides much less evidence. But being a first-stage retrieval method, `DeepCT-Index` has the potential to provide better candidates for these rerankers, hence improve the end-to-end search performance, as we will discuss next.

7.4.2 Impacts of `DeepCT-Index` on Downstream Rerankers

The second experiment examines whether an initial ranking produced by `DeepCT-Index BM25` can improve later-stage rerankers.

Table 7.4 reports the performance of two rerankers applied to candidate passages retrieved from `DeepCT-Index` and the *tf* index. The rerankers were selected based on their performance on the MS

Table 7.4: Re-Ranking accuracy of two rerankers applied to passages retrieved by BM25 from `DeepCT-Index` and the *tf* `index`. Dataset: MS-MARCO-Pas, dev query set.

	Recall		Conv-KNRM reranker MRR@10		BERT reranker MRR@10	
Depth	*tf*	DeepCT	*tf*	DeepCT	*tf*	DeepCT
10	40%	49%	0.234	0.270	0.279	0.320
20	49%	58%	0.244	0.277	0.309	0.343
50	60%	69%	0.253	0.278	0.336	0.361
100	68%	76%	0.256	0.274	0.349	0.368
200	75%	82%	0.256	0.269	0.358	0.370
500	82%	88%	0.256	0.269	0.366	0.374
1000	86%	91%	0.256 [1]	0.264	0.371 [1]	0.376

[1]The values are not exactly the same as in Table 7.3 due to differences in the initial rankings generated from our BM25 and the official BM25 baseline.

Table 7.5: Performance of competition systems using `DeepCT-Index` on the MS-MARCO-Pas dataset. Evaluation was done on dev set and hidden test set. Rankings were based on the accuracy on the test set on May 18, 2020. Statistical significance is unknown because the MS MARCO website publishes only summary results.

Method	dev	test	rank
BM25 + BERT reranker (Nogueira and Cho, 2019)	0.365	0.359	27
DeepCT + DocBERT reranker (*CMU*)	0.394	0.388	8
DeepCT + BART (*RMIT University*)	0.408	0.394	6
DeepCT + TF-Ranking BERT Ensemble (*Google Research*)	0.405	0.395	5
DeepCT + TABLE Model (*Meituan-Dianping*)	0.401	0.400	3

MARCO leaderboard (Table 7.3): Conv-KNRM (Dai et al., 2018), which has medium accuracy, and BERT reranker (Nogueira and Cho, 2019), which has high accuracy. We tested various re-ranking depths. Re-ranking at a shallower depth has higher efficiency but may miss more relevant passages.

The recall values show the percentage of relevant passages in the re-ranking passage set. `DeepCT-Index` had higher recall at all depths, meaning a ranking from `DeepCT-Index` provided more relevant passages to a reranker. The higher recall does lead to better end-to-end efficiency and/or effectiveness . Both rerankers consistently achieved higher MRR@10 by using `DeepCT-Index` compared to using *tf* index. For Conv-KNRM, the best MRR@10 improved from 0.256 to 0.278, and the required re-ranking depth decreased from 100 to 50. In other words, an initial ranking from `DeepCT-Index` helped the reranker to be over 8% more accurate and 2× more efficient. For BERT reranker, `DeepCT-Index` enabled the reranker to achieve similar accuracy using much fewer passages. Re-ranking the top 100-200 passages from `DeepCT-Index` produced similar MRR@10 as re-ranking the top 1,000 passages from *tf* index. The high computational cost of deep neural-based rerankers is one of the biggest concerns about adopting them in online services. `DeepCT-Index` reduces the re-ranking depth by 5× to 10×, making deep neural-based rerankers practical in latency-/resource-sensitive systems.

Besides the Conv-KNRM and BERT rerankers studied in this experiment, `DeepCT-Index` has also been adopted by several competition systems from several research groups. As shown in Table 7.5, 4 of

Table 7.6: Visualization of DeepCT-Index passage term weights. Red shades reflect the normalized term weights – the percentage of total passage term weights applied to the term. White indicates zero weight. Query terms are bold.

	Percentage of weights a term takes in the passage: 0% 10% 20% 30% 40% ¿50%
Query	**who is susan boyle**
On-Topic	Amateur vocalist **Susan** **Boyle** became an overnight sensation after appearing on the first round of 2009's popular U.K. reality show Britain's Got Talent.
Off-Topic	Best Answer: a troll is generally someone who tries to get attention by posting things everyone will disagree, like going to a **susan** **boyle** fan page and writing **susan** **boyle** is ugly on the wall. they are usually 14-16 year olds who crave attention.
Query	**what values do zoos serve**
On-Topic	**Zoos** **serve** several purposes depending on who you ask. 1) Park/Garden: Some **zoos** are similar to a botanical garden or city park. They give people living in crowded, noisy cities a place to walk through a beautiful, well maintained outdoor area. The animal exhibits create interesting scenery and make for a fun excursion.
Off-topic	There are NO purebred Bengal tigers in the U.S. The only purebred tigers in the U.S. are in AZA **zoos** and include 133 Amur (AKA Siberian), 73 Sumatran and 50 Malayan tigers in the Species Survival Plan. All other U.S. captive tigers are inbred and cross bred and do not **serve** any conservation **value**.
Query	**do atoms make up dna**
On-Topic	**DNA** only has 5 different **atoms** - carbon, hydrogen, oxygen, nitrogen and phosphorous. According to one estimation, there are about 204 billion **atoms** in each **DNA**.
Off-Topic	Genomics in Theory and Practice. What is Genomics. Genomics is a study of the genomes of organisms. It main task is to determine the entire sequence of **DNA** or the composition of the **atoms** that make up the **DNA** and the chemical bonds between the **DNA** **atoms**.

the top 10 systems on the MS MARCO Passage Ranking leaderboard were using DeepCT to generate the initial rankings at the time this dissertation was written. These results confirms the benefits of using DeepCT for pipelined ranking systems .

7.4.3 Sources of Effectiveness

This section aims to understand the sources of effectiveness of DeepCT-Index through several analyses.

Table 7.6 visualizes DeepCT-Index weights on a few cases. Each case has a query, a relevant passage, and a non-relevant passage that mentions query concepts but is actually off-topic. The heatmap visualizes the term weights in DeepCT-Index. Weights are normalized by the sum of all term weights in the passage, to reflect relative term importance in the passage.

DeepCT is able to identify terms that are central to the topic of the text. In the first query in Table 7.6, both passages mention *"susan boyle"*. In the relevant passage, DeepCT-Index recognized that the topic is *"susan boyle"* and puts almost all weight on these two terms. The off-topic passage is about the definition of *"troll"*, while *"susan boyle"* is used in an example. DeepCT-Index managed to identify the central concept *"troll"* and suppress other non-topical terms; less than 10% of weight goes to *"susan*

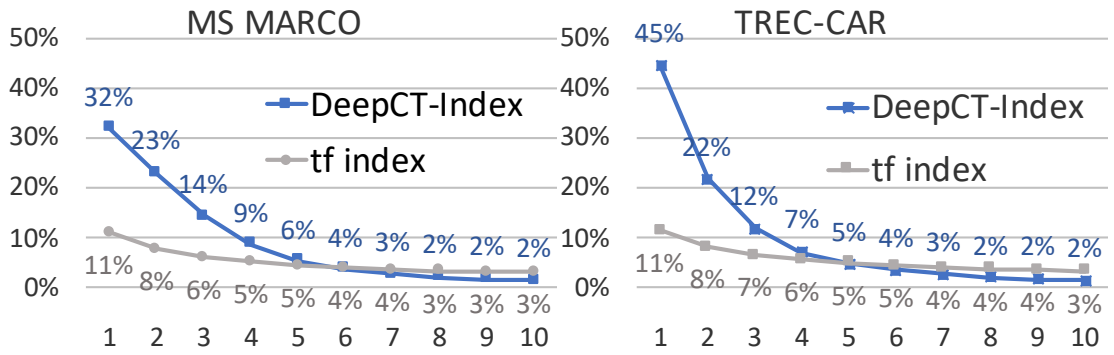


Figure 7.2: Term weight distribution of `DeepCT` weighted passages. It shows the weight distribution among the top-10 terms in passages with highest weights. The X-axis shows the term's rank ordered by weight. The Y-axis shows the average term weight normalized by total passage term weight.

boyle". With the new weights, the BM25 score between the query and the off-topic passage is greatly decreased. Similar behavior can be seen on other cases in Table 7.6.

Importantly, the examples in Table 7.6 shows that the term weights produced by `DeepCT-Index` are based on the meaning of the context rather than frequency. A term may get low weight even if it is frequent, e.g. in the last *"Genomics"* passage, *"DNA"* is considered unimportant even though it is mentioned 3 times. The same term receives very different weights in different passage even when *tf* is the same. This extent of independence from frequency signals is uncommon in previous term weighting approaches (Mihalcea and Tarau, 2004; Guo et al., 2016a; Zheng and Callan, 2015).

Figure 7.2 compares the term weight distribution of `DeepCT-Index` and *tf* index. It plots the average weight of each passage's highest-weighted term, the average weight of the second highest-weighted term, and so on. The original *tf* distribution is flat. `DeepCT-Index` assigns high weights to a few central terms, resulting in a skewed term weight distribution. Such skewed distribution confirms our observations from the case study: `DeepCT-Index` is capable of promoting a few central terms and suppressing off-topic terms.

7.5 Experimental Methodology and Results for DeepCT-Query

This last section presents experiment for the query term weighting task.

Experimental Methodology. The experimental methodology follows the settings used in (Zheng and Callan, 2015). We used the Robust04 and Gov2 datasets. Robust04 is a news corpus with 0.5M documents and 249 test topics. Gov2 is a web collection with 25M web pages and 150 test topics. Each test topic has 3 types of query: a short title query, a sentence-long query description, and a passage-long query narrative.

Experiments were done with 6 baselines that use two forms of query structure (`BOW`, `SDM`) and three types of term weights (`tf`, `DeepTR`, *Oracle*). `BOW` and `SDM` stand for bag-of-word queries and sequential dependency queries (Metzler and Croft, 2005). `tf` is the classic query term frequency weights. `DeepTR` is a previous state-of-the-art query weighting approach (Zheng and Callan, 2015). It extracts word features using the difference between the word's own embedding and the average embedding of the query. The features are linearly combined to produce term weights. `DeepTR` is supervised trained on term recall (Eq. 3). `Oracle` weights query terms by the ground truth term recall weights; it reflects how much `DeepTR` and `DeepCT-Query` can achieve if they made perfect predictions, estimating an upper limit.

Table 7.7: Retrieval accuracy of DeepCT-Query using QL retrieval model on Robust04 and Gov2. * indicates statistically significant improvements of BOW-DeepCT-Query over BOW-DeepTR. † indicates statistically significant improvements of SDM-DeepCT-Query over SDM-DeepTR. BOW-/SDM-*Oracle* weight terms by ground truth, estimating an upper limit for DeepTR and DeepCT-Query.

Query	Robust04					
	Title		Description		Narrative	
	NDCG@20	MAP	NDCG@20	MAP	NDCG@20	MAP
BOW-tf	**0.410**	**0.243**	0.417	0.248	0.320	0.177
BOW-DeepTR	0.381	0.226	0.427	0.264	0.330	0.183
BOW-DeepCT-Query	0.383	0.229	**0.445***	**0.273***	**0.367***	**0.213***
BOW-*Oracle*	*0.369*	*0.228*	*0.484*	*0.311*	*0.447*	*0.283*
SDM-tf	**0.427**	**0.264**	0.427	0.261	0.332	0.186
SDM-DeepTR	0.394	0.241	0.452	0.261	0.355	0.186
SDM-DeepCT-Query	0.394	0.242	**0.462**	**0.288**†	**0.380**†	**0.225**†
SDM-*Oracle*	*0.398*	*0.248*	*0.453*	*0.295*	*0.441*	*0.276*

Query	Gov2					
	Title		Description		Narrative	
	NDCG@20	MAP	NDCG@20	MAP	NDCG@20	MAP
BOW-tf	0.442	0.291	0.407	0.253	0.419	0.231
BOW-DeepTR	0.437	0.289	0.427	0.271	0.395	0.211
BOW-DeepCT-Query	**0.443**	**0.293**	**0.430**	**0.280**	**0.438***	**0.260***
BOW-Oracle	*0.453*	*0.306*	*0.462*	*0.312*	*0.481*	*0.298*
SDM-tf	**0.483**	**0.324**	0.434	0.270	0.436	0.239
SDM-DeepTR	0.482	0.324	**0.464**	**0.294**	0.432	0.237
SDM-DeepCT-Query	0.483	0.323	0.446	0.292	**0.455**†	**0.268**†
SDM-Oracle	*0.492*	*0.337*	*0.472*	*0.319*	*0.496*	*0.305*

Following Zheng and Callan (2015), we use the Indri search engine[10] with standard stemming and stop words filtering. We train and evaluate DeepCT-Query and DeepTR with 5-fold cross validation. Zheng and Callan (2015) show that query likelihood (QL) model performs slightly better than BM25, so we use QL to search the index. Retrieval results are evaluated using standard TREC metrics: NDCG@20 and MAP@1000. The BERT part of DeepCT was initialized with the official pre-trained BERT (uncased, base model) (Lee et al., 2019a). DeepCT, including the BERT layers and the last regression layer, was fine-tuned end-to-end. The model was trained for 10 epochs. We used a learning rate of $2e^{-5}$. Max input text length was set to be 30, 50 and 100 tokens for query titles, descriptions, and narratives.

Results. Results are listed in Table 7.7. The short title queries did not benefit from the term weighting approaches. Title queries often consist of a few keywords that are all essential, so re-weighting is less important. Besides, there isn't much context for DeepCT to leverage to estimate term importance. External information about the query, such as pseudo-relevance feedback (Lavrenko and Croft, 2001), may be necessary to understand short queries.

Description and narrative queries mention many terms and concepts; it is important to identify which are central during retrieval. In these cases, weighted queries are more effective than the un-weighted

[10] http://lemurproject.org/indri/

queries. DeepCT-Query is more accurate than DeepTR in most cases. DeepTR uses a relatively shallow method that represents the term by its word2vec (Mikolov et al., 2013) embedding, represents the entire query by the average of all its word embeddings, and adds contextual information to a term simply by subtracting the the average query embedding from the word embeddings. In contrast, DeepCT-Query uses BERT, a pre-trained 12-layer transformer model, to capture terms' context. The results demonstrate that DeepCT-Query better reflect a word's role in the query. Larger improvements were observed on narrative queries than on description queries. The results indicate that for short sentences, simple context modeling may be effective. But for more complex queries, a deep language modeling component like BERT can lead to improved search results.

7.6 DeepCT Summary

Recently, much research has focused on the reranking stages of multi-stage search engines. Most initial rankers are older-but-efficient bag-of-words retrieval models that use term frequency signals. However, frequency-based term weighting does not necessarily reflect a term's importance in queries and documents, especially when the frequency distribution is flat, such as in sentence-long queries or passage-long documents. More accurate term importance estimates require the system to understand the role each word plays in each specific context. This chapter presents DeepCT, a novel context-aware term weighting approach that better estimates term importance for first-stage bag-of-words retrieval systems.

The DeepCT framework is built on BERT. In BERT, a word's embedding gradually 'absorbs' information from other related words in the input text and generates a new word embedding that characterizes the word in a specific context. DeepCT uses BERT to extract contextual features and learns to use these features to predict the importance for each term in a supervised per-token regression task. The training signals are mined from relevance-based query-document pairs so that the predicted term weights are aligned with the retrieval task.

One use of DeepCT is DeepCT-Query, which weights query terms. Experimental results show that DeepCT-Query greatly improves the accuracy of longer queries, due to its ability to identify central query terms in a complex context.

A more novel use of DeepCT is DeepCT-Index, which weights document terms. DeepCT produces integer term weights that can be stored in a typical inverted index and are compatible with popular retrieval models such as BM25 and query likelihood. Experimental results show that DeepCT-Index improves the accuracy of two popular retrieval algorithms by up to 50%. Running BM25 on DeepCT-Index can be as effective as several previous state-of-the-art multi-stage search systems that use knowledge bases, machine learning, and large amounts of training data. The higher-quality ranking enabled by DeepCT-Index improves the accuracy/efficiency tradeoff for later-stage re-rankers. DeepCT was used by several research groups to generate initial rankings their rerankers, which led to top-ranked performance on a highly competitive benchmark.

Indexing and initial retrieval stages in IR have remained almost unchanged for several decades. Results from this chapter indicates that these frequency-based approaches are no longer sufficient. With recent advances in deep learning and NLP, it is time to revisit those indexing systems and retrieval models, moving from "frequencies" to "meanings". Instead of viewing the first stage retrieval as simple heuristics, DeepCT thinks of it as a trade off between efficient text representations (bag-of-words) and deep document understanding (BERT model), which opens up many new research possibilities. We believe our findings will inspire more innovations in deep retrieval systems in the near future.

Chapter 8

HDCT: Context-Aware Document Term Importance Estimation[1]

DeepCT (Chapter 7) demonstrates the potential of using neural networks to generate better bag-of-words text representations for efficient and effective retrieval. This chapter aims to broaden the applications of DeepCT in terms of the type of documents it can accept, the labels it can be trained on, and the retrieval models it supports.

One critical challenge of applying DeepCT to a wider range of retrieval tasks is the length constraint. Due to high memory complexity, BERT, a central component in DeepCT, is limited to short text of at most a few hundred words (Lee et al., 2019a). However, many search domains deal with longer documents. For example, in web search, some Wikipedia pages have over $10,000$ words. Those long documents cannot directly fit into DeepCT. Even if the memory allows, it is unknown if the model can effectively capture such long-term dependencies.

This chapter proposes HDCT, a context-aware hierarchical document term weighting framework. HDCT first estimates a term's local importance in each passage using DeepCT. Estimating at the passage level allows HDCT to handle long documents that exceed BERT's length limit (Lee et al., 2019a). Next, HDCT combines the local passage term weights into a global document bag-of-words representation. The document representation provides the retrieval model with both document-level and passage-level key terms, so that documents can be matched to queries accurately. These representations can be generated offline, stored in an inverted index, and retrieved efficiently with standard bag-of-words retrieval algorithms such as BM25.

Training HDCT requires having ground truth information about a term's importance in a passage. DeepCT mines the training labels from relevant query-document pairs, but such annotated data may not always be available. This paper proposes two weak-supervision strategies that automatically generate training labels from documents or pseudo-relevance feedback. The first strategy is a *content-based weak supervision* approach that solely relies on document contents. It exploits the internal structure of documents, and mines labels from certain document fields that were shown to provide a high-quality summary of the document (e.g., titles (Jin et al., 2002) and web inlinks (Eiron and McCurley, 2003)). The second strategy is a *PRF-based weak supervision* approach that trains HDCT on machine-generated pseudo-relevant feedback (PRF) labels. This is desgined for cases where user queries are available but not relevance signals, for example, if privacy regulations do not permit collecting user clicks.

The term weights produced by HDCT are stored into an inverted index, where the deep,passage-aware term weights replace the standard term frequency fields in the inverted lists. To retrieve documents, we

[1]This chapter is based in full on a previously published paper (Dai and Callan, 2020b) appearing in the Web Conference 2020.

follow DeepCT and use BM25 formula. In addition, we also investigates whether HDCT index is compatible with pseudo-relevance feedback retrieval techniques, a widely-used IR technique that estimates a query-specific term distribution based on retrieved documents.

Experiments show that the content-trained HDCT significantly improves bag-of-words retrieval models like BM25. It also works well with a pseudo-relevance feedback retrieval model RM3 (Lavrenko and Croft, 2001), and can be competitive with some supervised learning-to-rank pipelines. The PRF-based labels can bias document term weights towards people's search intents, leading to improved search accuracy. An analysis shows that BERT-based term weights are more effective than term frequencies at the passage level. The hierarchical document modeling approach successfully combines the passage-specific term weights for document retrieval, outperforming other approaches that combine passage retrieval scores.

Section 8.1 reviews related work. Sections 8.2 and 8.3 describe the HDCT framework and the three strategies to generate training labels. Experimental methodologies and results are presented in Sections 8.4 and 8.5. Section 8.6 concludes the chapter.

8.1 Related Work

This work is related to passage-based retrieval, and weak supervision in retrieval.

There is rich prior work using passages for document retrieval. As discussed in previously in Chapter 6.1, the most widely-used way to incorporate passage-level evidence is to *combine passage scores*. A large number of methods along this line of research have been proposed in the past few decades (Salton et al., 1993; Kaszkiel and Zobel, 1997, 2001; Liu and Croft, 2002; Wu et al., 2019). This work falls into a different category that *combines passage representations*, which uses passage-level term statistics to build document representations, and evaluates queries on the document representations. This research approach has received less attention. The most related work to ours is from Catena et al. (2019) where the authors models document term weights using a weighted sum of term frequencies per passage based on position of the passage in the document. It is an open question how to go beyond these simple statistics and mine deeper signals from passages to better represent documents.

IR research mainly focuses on two types of weak supervision signals for training models: *content-based* signals and *pseudo-relevance feedback* based signals. Content-based approaches are motivated by the observation that the document content often exhibits some relevance relations between pieces of text. Research on this topic dates back at least 20 years. For example, Berger and Lafferty (1999) trains statistical machine translation models by generating synthetic queries using mutual information in the documents. Jin et al. (2002) used title-document pairs to train statistical translation models. Asadi et al. (2011) used web anchor text (inlinks) to generate pseudo labels to train learning-to-rank models. Recently, Berendsen et al. (2013) used hashtags for training and tuning Microblog rankers. More recently, MacAvaney et al. (2019b) revisited this topic to train neural ranking models, and showed that training neural ranking models with document titles as pseudo queries can outperform unsupervised BM25. Pseudo-relevance feedback (PRF) based approaches make use of the rankings from a search engine to generate pseudo-relevant labels. (Zamani and Croft, 2017) trains word embeddings using queries and their top ranked documents returned by the query likelihood retrieval model, and uses the embeddings to enhance the query likelihood retrieval model. (Dehghani et al., 2017) uses queries and documents retrieved by BM25 to train a neural reranker. This approach was further used in (Zamani et al., 2018a) to learn documents embeddings for the initial retrieval stage. In general, recent research on both of the above two types of weak supervision focused on embeddings (Zamani and Croft, 2017; Zamani et al., 2018a) or non-BERT neural ranking models (Dehghani et al., 2017; MacAvaney et al., 2019b). Their effectiveness in learning discrete bag-of-words document representations with BERT-based models remains to be studied.

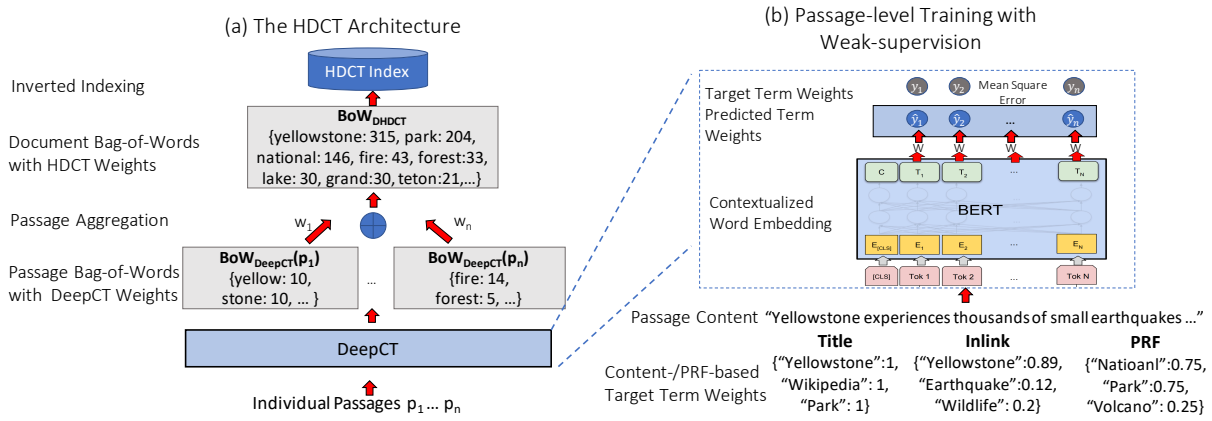


Figure 8.1: The HDCT architecture.

8.2 The HDCT framework

This section presents the hierarchical document term weighting framework, HDCT, as shown in Figure 8.1(a).

Given a document d, HDCT estimates passage-level term weights using contextual term representations generated by DeepCT. Next, HDCT combines the passage-level term weights into document-level term weights. The output is a document bag-of-words representation that can be stored in a standard inverted index and retrieved by common bag-of-words retrieval models like BM25 and pseudo relevance feedback retrieval models like RM3 (Lavrenko and Croft, 2001).

8.2.1 Passage-Level Term Weighting

Given a document d, HDCT first splits it into a sequence of passages $P_d = \{p_1, ..., p_n\}$. The maximum input text length of BERT is 512 tokens after tokenization – about 300 to 400 English words before tokenization. Meanwhile, prior research shows that fixed-size passages of 200-300 words more are effective than natural passages (Kaszkiel and Zobel, 1997). Therefore, passages in HDCT consist of consecutive sentences that make up to about 300 words.

Next, HDCT estimates term importance in each passage using the context-aware term weighting framework DeepCT (Chapter 7). Figure 8.1(b) shows the details of this step. DeepCT uses BERT to generate contextualized term embeddings, and project each term's embedding into a real-valued number $\hat{y}_{t,p_i}$, which reflects the term t's importance in the current passage p_i.

The original DeepCT (Chapter 7) scales $\hat{y}_{t,p_i}$ into a *tf*-like integer that can be used with existing retrieval models by applying a linear scaling function:

$$tf_{\texttt{DeepCT-Sqrt}}(t, p_i) = round(N * \hat{y}_{t,p_i}), \tag{8.1}$$

where $\hat{y}_{t,p}$ is the predicted weight. N scales the weight into a integer range, e.g. $N = 100$ keeps two digit precision.

In this work, we use a modified version of the scaling function that takes the square-root of $\hat{y}_{t,p_i}$:

$$tf_{\texttt{DeepCT}}(t, p_i) = round(N * \sqrt{\hat{y}_{t,p_i}}), \tag{8.2}$$

The square-root function is used for smoothing. As shown in our previously analysis (Chapter 7.4), the term weights from DeepCT are very skewed – most terms have very low weight and will be ignored by the retrieval model. With square-root function, low predicted weights will be raised up, e.g. $\sqrt{0.01} = 0.1$, preventing the document representation from being dominated by a few highly-weighted terms.

After the scaling and post-processing, we generate a bag-of-words vector representation of the passage p:

$$\text{P-BoW}_{\text{HDCT}}(p_i) = [tf_{\texttt{DeepCT-Sqrt}}(t_1, p_i), .., tf_{\texttt{DeepCT-Sqrt}}(t_m, p_i)]. \tag{8.3}$$

Its words are from the original text of the passage; the weights are a *tf*-like integer based on predictions from DeepCT.

The above steps are applied to every passage $p_1, ..., p_n$ in the document d. At the end, HDCT generates a sequence of bag-of-words passage vectors.

$$\{\text{P-BoW}_{\text{HDCT}}(p_1), ..., \text{P-BoW}_{\text{HDCT}}(p_n)\}. \tag{8.4}$$

8.2.2 Document-Level Term Weighting

The previous step generates a sequence of passage bag-of-words representations. The next question is how to combine the passages for document retrieval.

A widely-used approach is to index and retrieve the passages independently, and aggregate passage scores at query time. This approach is widely used in prior research (Salton et al., 1993; Kaszkiel and Zobel, 1997, 2001; Liu and Croft, 2002; Wu et al., 2019) as well as our previous research on the DocBERT reranker (Chapter 6). However, passage-level retrieval often faces the challenge of lacking document-level context (Wu et al., 2019). Though in Chapter 6 we use document titles to provide global context for passages, that work focused on reranking. In the initial retrieval stage, titles may not be sufficient as the retrieval models are much simpler and the document sets are much larger. So far, results on combining passage scores for initial retrieval were mixed (Liu and Croft, 2002; Wu et al., 2019), and the common practice is to use document level bag-of-words representations.

As shown in Figure 8.1(a), HDCT seeks to aggregate passage representations into document-level ones for document retrieval. A term's importance in the document is a weighted sum of its importance in each passage:

$$\text{D-BoW}_{\text{HDCT}}(d) = \sum_{i=1}^{n} pw_i \times \text{P-BoW}_{\text{HDCT}}(p_i). \tag{8.5}$$

pw_i models the importance of the i-th passage p_i to the document d. This work explores two options for determining pw_i. The first one uses $pw_i = 1$ (sum); it weights all passages equally. The second one uses $pw_i = \frac{1}{i}$ (decay); it discounts passages based on the position, as prior research found that passages at the beginning of a document tend to attract more attention from readers and are more important for relevance estimation (Wu et al., 2019; Catena et al., 2019). Following Wu et al. (2019), we use the reciprocal of position as the weight of a passage.

Besides passage position, it is also intuitive to weight passages from their content. This work does not explicitly model this factor. But as discussed in the next section, we can train HDCT to down-weight all terms in a passage, hence implicitly weight passages based on content.

In most prior work, passages only provide limited signals for bag-of-words retrieval, such as passage term frequencies or passage positions. HDCT uses BERT to extracts much richer evidence from passages, leveraging the deep content understanding of passages to build the document bag-of-words.

8.2.3 Retrieval with HDCT Index

HDCT stores BoW_{HDCT}, the document bag-of-words representation, into an inverted index, where the new context-aware term weights replace the standard term frequency fields in the inverted lists (Croft et al., 2009). We call it the HDCT index. In terms of efficiency, HDCT does not introduce new words into documents, so the index does not become larger. Usually, HDCT reduces the index size as some terms' weight becomes 0 during the scaling in Eq. 8.2.

To retrieve documents from HDCT indexes, we follow DeepCT and use the standard BM25 formula. During retrieval, the term frequency value in BM25 is replaced with the term weights stored in the HDCT index. HDCT is expected to improve retrieval by identifying key terms in a document.

This work also investigates whether HDCT index is compatible with pseudo-relevance feedback retrieval algorithms. In particular, we study the BM25+RM3 model for its strong performance shown in prior work (Lin, 2019).

Let $D_R = \{d_1, d_2, ..., d_k\}$ be the top k documents retrieved from a HDCT index in response to the query using the standard BM25. An expanded query is generated using RM3 (Lavrenko and Croft, 2001):

$$P_{RM3}(t, q|R) = (1 - \lambda)P(t|q) + \lambda \sum_{d \in D_R} P(t|d)P(q|d), \tag{8.6}$$

where $P(t|d)$ is estimated with the $\text{D-BoW}_{\text{HDCT}}(d, t)$ from Eq. 8.5:

$$P(t|d) = \frac{\text{D-BoW}_{\text{HDCT}}(d, t)}{\sum_{t_i} \text{D-BoW}_{\text{HDCT}}(d, t_i)}. \tag{8.7}$$

We then run a second round of retrieval using the expanded query. The retrieval is again performed on the HDCT index.

8.3 Training Strategies For HDCT

Similar to DeepCT, HDCT is trained on a passage-level per-token regression task. It is expensive to manually label the importance of every token in every passage ($y_{t,p}$). To automatically generate labels, the key question is, *what evidence do we have that shows a term's importance for document retrieval?*. This chapter exploits document content and user search data, and proposes three training strategies: a relevance-based approach for cases where rich query-document relevance assessments are available, a PRF-based approach for cases where search queries can be collected but the relevance labels or user activities are not accessible, and a content-based approach for cases where only the documents are available.

8.3.1 Supervision from Relevance and Pseudo-Relevance Feedback

As shown in Figure 1(b), in HDCT, only the BERT component and the linear layer parameters w, b need to be trained. Theoretically, they can be trained on the document level. In practice, this leads to complex implementations to handle memory limitations and the uncertain number of passages. For simplicity, we train them on the passage-level – the model considers a single passage at a time. This simplification makes HDCT training objective the same as DeepCT (Chapter 7.2). Given the target term weight (ground truth) for a term t in a passage p, denoted as $y_{t,p}$. HDCT minimizes the mean square error between the predicted weights $\hat{y}$ and the target weights y:

$$MSE = \sum_{p} \sum_{t \in p} (y_{t,p} - \hat{y}_{t,p})^2. \tag{8.8}$$

Given a training document d, its passages $P_d = \{p_1, ..., p_n\}$, and its relevant queries $Q_d = \{q_1, ..., q_b\}$, we generate the *relevance-based* training labels for each passage using the query term recall weights described in Chapter 7.2.

$$y_{t,p} = \frac{|Q_{d,t}|}{|Q_d|} \in [0, 1]. \tag{8.9}$$

t is a term from passage p. $Q_{d,t}$ is the subset of queries that mention term t. If a document d is relevant to many queries that mention the word t, then t is likely to be essential for this document.

As shown in Eq. 8.9, the training labels $y_{t,p}$ are passage independent and only depend on the document's relevant queries Q_d. That means, if 70% of the document's queries mention the term *"yellowstone"*, then the target term-weights for *"yellowstone"* will always be 0.7 regardless of which passage it appears in. Section 8.3.3 discusses the effects of such global labels in detail.

In some cases, the search queries are available, but the relevance feedback signals, such as clicks, are not available (for example, in privacy-sensitive scenarios). Inspired by Zamani et al. (2018a), we propose a *pseudo-relevance based (PRF-based)* weak supervision strategy for `HDCT`.

In the PRF-based weak supervision strategy, an existing retrieval system, e.g., standard `BM25`, is used to retrieve potential relevant documents for each query. Each document's pseudo-relevant queries, PRF-Q_d, are collected to generate the PRF-based training labels using the same procedure as Eq. 8.9:

$$y_{t,p} = \frac{|\text{PRF-}Q_{d,t}|}{|\text{PRF-}Q_d|} \in [0, 1]. \tag{8.10}$$

8.3.2 Supervision from the Content

A generally applicable search system should be able to build a good search engine just from the document collection. Towards this goal, the third training strategy mines labels from the documents themselves.

In many domains, the documents are loosely structured with various sources of textual information (fields), such as title, keywords, and inlinks (anchor text from documents that link to this document). Various researches have shown that these fields behave like real user queries (Eiron and McCurley, 2003; Jin et al., 2002). They provide a short summary of what a document is about and which search intents it may satisfy. These short, highly representative fields provide evidence about which terms bear high importance in the document.

Let F_d be a reference field that we use to train `HDCT`, e.g., the inlink field. We denote $F = \{f_1, ..., f_m\}$, where each element f_i is a text instance of the reference field. Some fields only have a single instance, e.g., a document usually has one title. Some fields may have multiple instances, e.g., a web page can have thousands of inlinks. The *content-based* strategy mines weak supervision signal about a term's importance by checking if, and how frequently, the term appears in the reference field.

Formally, given a training document d, its passages $\{p_1, ..., p_n\}$, and its reference field $F_d = \{f_1, ..., f_n\}$, the content-based weak-supervision approach generates labels as the following:

$$y_{t,p} = \frac{|F_{d,t}|}{|F_d|} \in [0, 1], \tag{8.11}$$

where t is token from passage p, and $\frac{|F_{d,t}|}{|F_d|}$ is percentage of field instances that contain t. When there is a single instance, e.g., a document title, Eq. 8.11 generates a binary label indicating if term t appears in the field or not. When there are multiple instance, e.g. inlinks, Eq. 8.11 is a real number between 0 and 1. In the latter case, a token is considered more important if it is mentioned by a large portion field instances, reflecting the "collective wisdom".

Similar to the relevance-based and PRF-based strategies, the training labels $y_{t,p}$ are actually passage independent and only depend on the document's reference field F_d. For example, if the document's title is *"Yellowstone National Park"*, then the target term-weights for *"yellowstone"* will always be 1 regardless of which passage it appears in.

The training passage p and its target term weights derived from the reference field, are used to train HDCT by minimizing the MSE loss in Eq. 8.8. These training labels are automatically extracted from the document content; no task-specific data collecting or labeling is required.

8.3.3 Global Labels for Local Term Weighting

In the above three approaches, the target term weights (labels) are derived globally from the entire document, rather than being locally dependent on specific passages. One would expect these global labels to be less effective for passage term weighting. However, in practice, as BERT's contextualized word representation always varies with the passage, HDCT can still generate local term weights even though the training labels are global.

Moreover, these global labels let HDCT capture the global importance of passages. Some passages introduce noise, e.g., advertisements, navigation bars, or large blocks of equations. These passages do have their own locally important words, but they should not have high weights in the document. The document-derived labels teach HDCT to down-weight the entire passage. For example, low-quality passages often do not cover any inlink terms. Rather than trying to find the locally important words, HDCT gives 0 weight to all words in these passages. As a result, the entire passage makes little contribution to the document bag-of-words representation. We will illustrate the passage-weighting effect in Section 8.5.3.

8.4 Experimental Methodologies

This section presents our experimental methodologies, including datasets, baselines, and experimental methods.

8.4.1 Datasets

Experimental evaluation of HDCT used 4 *document retrieval* datasets with different characteristics. The documents in these datasets are longer than the *passage retrieval* datasets used in the DeepCT study (Chapter 7), allowing us to test how well HDCT supports long documents.

The first data set is ClueWeb09-B, a widely used text retrieval collection. The original collection contains 50 million web pages; we used the spam-filtered subset of 33 million documents. Spam was filtered using the Waterloo spam score (Cormack et al., 2011) with a threshold of 60. The documents were split into a total of 100 million passages using a non-overlapping window of around 300 words. A document consists of 4 fields: title, URL, inlinks, and body.

The second dataset is ClueWeb09-C. Running HDCT over ClueWeb09-B is time consuming, making it slow to experiment with different model configurations. Therefore, we created ClueWeb09-C, a 10% subset of the original corpus. It consists of a 10% random sample of ClueWeb09-B documents, plus all documents that were in the original TREC judgment pool (in the qrels files)[2]. In total, there are 3.4 million documents and 10 million passages.

The third dataset is ClueWeb12-C. ClueWeb12-B13 is also a standard text retrieval collection used in IR research. We created the 10% subset, called ClueWeb12-C, using the same method described above.

[2]If not included, many queries ended up with few or no relevant documents, making the evaluation results unstable.

Spam filter was not applied as suggested in (Dalton et al., 2014). In total, there are 5 million documents and 13 million passages. A document consists of four fields (title, URL, inlinks, and body).

The 2009-2014 TREC Web Tracks provided 200 queries with relevance assessments for ClueWeb09, and 100 queries for ClueWeb12. They were used for evaluating HDCT. Two versions of queries were evaluated: a short keyword query (title query) and a longer natural language query (description query). Evaluation used NDCG@20, the main metric for the TREC Web Tracks; MAP@1000, to show the effectiveness at deeper rankings; and MRR, to be consistent with MS-MARCO-Doc.

The fourth dataset is the MS-MARCO Document Ranking dataset (MS-MARCO-Doc)[3]. It is is a benchmark dataset for web document retrieval recently released in the TREC 2019 Deep Learning Track. The dataset shares the similar corpus and queries with the MS-MARCO Passage Ranking (MS-MARCO-Pas) dataset used in Chapter 7, but MS-MARCO-Doc aims to retrieve documents rather than passages. The dataset has 4 million documents, which produced 12 million passages. A document consists of 3 fields (title, URL, and body). The dataset comes with a training set of 0.37 million queries and the corresponding relevant documents. Evaluation was on the dev set, which contains 5193 queries. Evaluation used the mean reciprocal rank (MRR) as suggested in the official instructions (Nguyen et al., 2016).

8.4.2 Experimental HDCT Methods

On the ClueWeb datasets (ClueWeb09-B/C and ClueWeb12-C), we tested three experimental HDCT methods trained with various supervision strategies:

- HDCT-title was trained with the content-based weak supervision strategy, using titles as the reference field. It generated term weights for every document in the collection, and built an inverted index for retrieval over the collection.
- HDCT-inlink was trained with the content-based weak supervised strategy, using inlinks as the reference field. We removed URL inlinks, and the most frequent inlinks like "next page" that has been linked to more than $10,000$ documents in the collection.
- HDCT-PRFaol was trained with the PRF-based weak supervision strategy using the AOL query log (Pass et al., 2006) and pseudo-relevance labels. We removed queries that are URLs and the 100 most frequent ones. We randomly sampled 500K unique queries, which is about the same scale as the MS-MARCO-Doc training query set used in HDCT-PRFmarco. It was also of the same scale as used in prior research (MacAvaney et al., 2019b). As recommended by MacAvaney et al. (2019b), 10 documents were retrieved for each query using BM25FE, a strong baseline that ensembles BM25 scores on each field; we sampled 1 from the top 10 documents to reduce computational cost.

Four HDCT methods were tested on MS-MARCO-Doc:

- HDCT-title was trained with document titles. MS-MARCO-Doc does not have inlink data, so HDCT-inlink is not available.
- HDCT-PRFaol was trained with the PRF-based strategy using the AOL query log and pseudo-relevance labels.
- HDCT-PRFmarco was trained with the PRF-based weak supervision strategy using the MS-MARCO-Doc training queries and pseudo-relevance labels. Same as HDCT-PRFaol, we retrieved top 10 documents for each query using BM25FE and sample 1 to generate the training data. HDCT-PRFaol and HDCT-PRFmarco allows us to compare the differences between out-of-domain and in-domain queries.

[3] https://microsoft.github.io/TREC-2019-Deep-Learning/

- `HDCT-supervised` was a fully-supervised model trained with the relevance-based supervision. It used 0.37 million relevance assessments in the MS-MAROC-Doc training set.

All models were trained for 100K steps with a batch size of 16 and a learning rate of $2e-5$; training over 100K steps did not lead to significant improvements. BERT parameters was initialized with the official pre-trained BERT (uncased, base model) (Tenney et al., 2019a). Maxium input length was set to 512 tokens. The scaling coefficient N in Eq (8.2) and the passage weights pw in Eq (8.5) were selected based on the dataset. We chose N from $\{10, 100\}$, and the passage weights from $\{\texttt{sum}, \texttt{decay}\}$. Unless otherwise specified, the rest of this chapter reports the best configuration of each dataset, that is $N = 10$ with `sum` for ClueWeb datasets, and $N = 100$ with `decay` for MS-MARCO-Doc.

We used Lucene to build the indexes using the term weights from `HDCT` methods. The indexes were tested with three widely-used retrieval models.

- `BM25`. The BM25 retrieval model (Robertson and Walker, 1994) is a widely-used well-performing bag-of-words retrieval model.
- `BM25FE`. `BM25FE` is an ensemble of `BM25` rankers on different document fields. Field scores are linearly combined in the ensemble, where the weights are found through a parameter sweep. ClueWeb datasets used title, URL, inlink, and body. MS-MARCO-Doc used title, URL and body; inlink was not available in this dataset. `HDCT` only weighted terms in the body field.
- `BM25+RM3`. The relevance model RM3 (Lavrenko and Croft, 2001) is a popular query expansion technique using pseudo-relevance feedback. `BM25+RM3` has been shown to improve the original `BM25`, and has been considered a strong baseline. We tested the compatibility between `HDCT` index and `BM25+RM3` as described in Section 8.2.3

We used the Anserini (Yang et al., 2017) implementation of the above retrieval models. We *tuned the parameters* of these retrieval models on the evaluation query sets through 2-fold cross-validation. These include: the k_1 and b parameters in `BM25`, the field weights in `BM25FE`, and the number of feedback documents, the number of feedback terms, and the feedback coefficient in `BM25+RM3`.

8.4.3 Baselines

The main baseline is *tf*, the standard term frequency based document index, e.g., as used by Luccene and Indri. We used three strong bag-of-words retrieval models on the *tf* index, including `BM25`, `BM25FE`, and `BM25+RM3`.

We also compared `HDCT`, which uses discrete bag-of-words, to two retrieval models that use embeddings: `RLM` (Zamani and Croft, 2017) and `SNRM` (Zamani et al., 2018a). Same as `HDCT`, they support efficient full-collection retrieval. `RLM` (Zamani and Croft, 2017) makes use of word embedding similarities for pseudo-relevance feedback. `SNRM` (Zamani et al., 2018a) is the current state-of-the-art embedding-based index. It converts documents into sparse $20,000$-dimension embeddings, and store them in an inverted index. `RLM` and `SNRM` are both trained using a PRF-based weak supervision approach. The authors did not release the trained models or indexes, and we were not able to fully optimize our own implementations due to the large amount of training data they require [4]. Therefore, we report the results reported by the authors on the ClueWeb09-B dataset.

Finally, we compared `HDCT` to two strong reranking systems, which are more computationally complex and require training data. The first, `Coor-Ascent`, is a strong feature-based learning-to-rank method using Coordinate Ascent; it was also used as baselines in Chapters 4 and 6. The second,

[4] `RLM` and `SNRM` used 6 million queries and 6×10^7 to 6×10^{13} training examples (Zamani and Croft, 2017; Zamani et al., 2018a).

Table 8.1: Effectiveness of content-trained HDCT indexes on the ClueWeb09-C dataset. $*$ indicates statistically significant improvements over *tf*, the standard inverted index using term frequency.

ClueWeb09-C		Title Query					
Retrieval Model	Indexing Term Weight	MRR		NDCG@20		MAP@1000	
BM25	*tf*	0.493	–	0.307	–	0.248	–
	HDCT-title	0.592*	20%	0.342*	11%	0.254	3%
	HDCT-inlink	0.586*	19%	0.356*	16%	0.265*	7%
BM25FE	*tf*	0.591	–	0.322	–	0.250	–
	HDCT-title	0.604	2%	0.358*	11%	0.263*	5%
	HDCT-inlink	0.615	4%	0.361*	12%	0.270*	8%
BM25+RM3	*tf*	0.563	–	0.350	–	0.278	–
	HDCT-title	0.610*	8%	0.369*	6%	0.280	1%
	HDCT-inlink	**0.630***	12%	**0.397***	14%	**0.298***	7%

ClueWeb09-C		Description Query					
Retrieval Model	Indexing Term Weight	MRR		NDCG@20		MAP@1000	
BM25	*tf*	0.570	–	0.321	–	0.238	–
	HDCT-title	0.608	7%	0.362*	13%	0.257*	8%
	HDCT-inlink	0.625	9%	0.377*	17%	0.264*	11%
BM25FE	*tf*	0.651	–	0.357	–	0.269	–
	HDCT-title	0.663	2%	0.376*	5%	0.274	2%
	HDCT-inlink	0.643	-1%	0.385*	8%	0.280*	4%
BM25+RM3	*tf*	0.581	–	0.351	–	0.257	–
	HDCT-title	0.634*	9%	0.386*	10%	0.276*	7%
	HDCT-inlink	**0.663***	14%	**0.399***	14%	**0.285***	11%

DocBERT-FirstP, is our BERT-based re-ranking method described in Chapter 6. Both methods adopt the settings used in Chapters 4 and 6 and re-ranked the top 100 documents retrieved by Galago SDM. Code and data for HDCT can be found in our virtual appendices. [5]

8.5 Experimental Results

Three experiments were conducted to study: the retrieval effectiveness of content-trained HDCT; the effects of stronger supervision using relevance-based and PRF-based labels; and the effects of different types of hierarchical document modeling.

8.5.1 Performance of Content-Trained HDCT

When building a search system for a new document collection, it is often the case that there are no relevance labels to train machine learning models. Typically, people would build a *tf*-based inverted index

[5] http://boston.lti.cs.cmu.edu/appendices/TheWebConf2020-Zhuyun-Dai/

Table 8.2: Effectiveness of content-trained HDCT indexes on the ClueWeb12-C dataset. $*$ indicates statistically significant improvements over *tf*, the standard inverted index using term frequency.

ClueWeb12-C		Title Query					
Retrieval Model	Indexing Term Weight	MRR		NDCG@20		MAP@1000	
BM25	*tf*	0.545	–	0.211	–	0.050	–
	HDCT-title	0.607*	11%	0.230*	9%	0.054*	8%
	HDCT-inlink	0.603	10%	0.232*	10%	0.055*	11%
BM25FE	*tf*	0.584	–	0.229	–	0.054	–
	HDCT-title	0.611	5%	0.236	3%	0.058*	6%
	HDCT-inlink	0.613*	5%	**0.241***	5%	**0.060***	11%
BM25+RM3	*tf*	0.567	–	0.216	–	0.051	–
	HDCT-title	**0.642***	13%	0.235*	9%	0.056*	10%
	HDCT-inlink	0.622*	10%	0.241*	12%	0.058*	11%

ClueWeb12-C		Description Query					
Retrieval Model	Indexing Term Weight	MRR		NDCG@20		MAP@1000	
BM25	*tf*	0.535	–	0.183	–	0.043	–
	HDCT-title	0.621*	18%	0.218*	19%	0.053*	22%
	HDCT-inlink	0.602*	13%	0.215*	17%	0.052*	19%
BM25FE	*tf*	0.554	–	0.197	–	0.048	–
	HDCT-title	0.631*	12%	0.218*	11%	0.053*	9%
	HDCT-inlink	0.619*	12%	0.217*	10%	0.053*	10%
BM25+RM3	*tf*	0.503	–	0.186	–	0.043	–
	HDCT-title	**0.635***	26%	**0.221***	19%	**0.054***	25%
	HDCT-inlink	0.610*	21%	0.220*	19%	0.053*	21%

Table 8.3: Effectiveness of content-trained HDCT indexes on MS-MARCO-Doc. $*$: statistically significant improvements over *tf*, the standard inverted index using term frequency.

MS-MARCO-Doc		Dev Queries	
Retrieval Model	Indexing Term Weight	MRR	
BM25	*tf*	0.254	–
	HDCT-title	0.287*	13%
BM25FE	*tf*	0.283	–
	HDCT-title	0.300*	6%
BM25+RM3	*tf*	0.250	–
	HDCT-title	0.288*	15%

and use out-of-the-box retrieval models like BM25. Our goal is to construct a better index using content-trained HDCT without relying on any additional labels. The first experiment tests if content-trained HDCT

Table 8.4: Effectiveness of HDCT(-inlink) on the ClueWeb09-B dataset. We report MAP@100 because Coor-Ascent and DocBERT-FirstP re-ranked the top 100 documents. Superscripts 1-8 indicate statistically significant improvements over the corresponding methods, as labeled in the first column. For example, ^{14} means that this result is statistically significantly better than runs 1 and 4.

ClueWeb09-B	Query Title					
Method	MRR		NDCG@20		MAP@100	
1 BM25, *tf*	0.477	-12%	0.272	-8%	0.154	-4%
2 BM25FE, *tf*	0.530^{1}	-2%	0.268	-9%	0.157	-3%
3 BM25+RM3, *tf*	0.520^{1}	-4%	0.294^{12}	-0%	0.164 ^{1}	+2%
4 Coor-Ascent	0.500	–	0.295^{12}	–	0.161^{1}	–
5 DocBERT-FirstP	0.538^{14}	-1%	0.286^{12}	-3%	0.166^{12}	+3%
6 BM25, HDCT	0.543^{1234}	+0%	0.303^{1235}	+3%	0.163^{1}	+1%
7 BM25FE, HDCT	0.543^{1234}	+0%	0.303^{12345}	+3%	0.163^{12}	+1%
8 BM25+RM3,HDCT	**0.597**$^{1-7}$	+10%	**0.326**$^{1-7}$	+11%	**0.180**$^{1-7}$	+12%

ClueWeb09-B	Query Description					
Method	MRR		NDCG@20		MAP@100	
1 BM25, *tf*	0.471	-6%	0.234	-7%	0.134	-7%
2 BM25FE, *tf*	0.511^{13}	3%	0.250^{1}	-0%	0.139^{1}	-4%
3 BM25+RM3, *tf*	0.473	-6%	0.249^{1}	-1%	0.138	-5%
4 Coor-Ascent	0.503^{12}	–	0.251^{1}	–	0.145^{123}	–
5 DocBERT-FirstP	**0.532**^{1236}	+6%	0.272^{1234}	+8%	**0.151**^{1236}	+4%
6 BM25, HDCT	0.510^{13}	+1%	0.267^{1234}	+6%	0.143^{13}	-1%
7 BM25FE, HDCT	0.521^{1234}	+3%	0.271^{1234}	+8%	0.145^{123}	+0%
8 BM25+RM3,HDCT	0.525^{12346}	+4%	**0.274**^{12346}	+9%	0.148^{123}	+2%

can outperform standard *tf*-based retrieval models, strong supervised re-ranking models, and competitive embedding-based retrieval models.

Comparison to Standard *tf* Index. Tables 8.1-8.3 show the retrieval effectiveness of several content-trained HDCT indexes on the ClueWeb09-C, ClueWeb12-C and MS-MARCO-Doc datasets. HDCT-title and HDCT-inlink use document title/inlinks as the reference field to generate training labels. The baseline is a typical term-frequency (*tf*) based inverted index. Significant and robust gains from HDCT over *tf* were observed on all datasets and query sets under various retrieval models.

When BM25 was used, HDCT indexes were 10%-20% more accurate than the *tf* index.It shows that HDCT weights are more effective than simply counting term frequencies in the document.

When BM25FE was used, the gap between HDCT and *tf* was smaller, but HDCT still outperformed *tf* in most cases. It shows that the content-trained HDCT can provide new information not covered by titles and inlinks. Titles and inlinks are often short and incomplete. Sometimes they have low text quality. Learning from a large number of titles/inlinks of different styles and qualities helps HDCT to capture general patterns of term importance, generating smoother and cleaner term weights than the original text fields.

RM3 is a pseudo-relevance feedback retrieval model originally designed for *tf* weights. Our results show that HDCT weights also fits into RM3. HDCT brought significantly improvements to the original *tf*-based BM25+RM3. The combination of HDCT-inlink index and BM25+RM3 retrieval achieved the

Table 8.5: Comparison between HDCT and embedding-based retrieval models. Dataset: ClueWeb09-B, title queries. Results for RLM are from (Zamani and Croft, 2017); results for SNRM are from (Zamani et al., 2018a). HDCT were evaluated using the methodology in (Zamani et al., 2018a; Zamani and Croft, 2017) to be directly comparable.

	RLM (Zamani and Croft, 2017)		SNRM (Zamani et al., 2018a)		HDCT-inlink	
tf Baseline	0.224	–	0.229	–	0.246	–
Retrieval	–	–	0.235	+3%	0.270	+10%
Retrieval+PRF	0.236	+5%	0.248	+8%	0.289	+17%

best accuracy on ClueWeb09-C. Its NDCG@20 has a 14% improvement from *tf*-based BM25+RM3, 22% from the the *tf*-based BM25FE, and 29% from the *tf*-based BM25.

Comparison to Supervised Re-ranking Systems. Next, we test HDCT on the ClueWeb09-B dataset. ClueWeb09-B is a standard test collection widely used in IR research. It allowed us to compare HDCT to other published results. Table 8.4 compares HDCT-inlink, the best model found on ClueWeb09-C, to the standard *tf* retrieval models and two supervised re-ranking systems: LeToR and BERT-FirstP. They were trained on around $10,000$ relevant query-document pairs. HDCT-inlink was not trained on any relevance labels.

HDCT-inlink outperformed the standard *tf* on ClueWeb09-B, which has $10\times$ irrelevant documents as ClueWeb09-C. It demonstrates the robustness of HDCT to larger and noisier collections.

HDCT-inlink was also as good as or better than some of the re-ranking systems. On query titles, retrieval using BM25+RM3 from HDCT-inlink was significantly more accurate than both re-rankers. On query descriptions, it outperformed LeToR and was on-par with BERT-FirstP at the top of the ranking. ClueWeb09-B is a realistic condition in which only a moderate amount of training data is available. These results indicate that under such conditions, HDCT can be very competitive with some state-of-the-art supervised re-ranking pipelines without using any relevance labels.

In terms of efficiency, re-ranking models like BERT-FirstP are computationally expensive at the query time. HDCT index is built offline and uses simple bag-of-words retrieval at query time, making it preferable in efficiency-sensitive scenarios.

Comparison to Embedding-Based Retrieval. HDCT only uses exact lexical term matching signals for retrieval. Embedding-based retrieval models, on the other hand, can soft match text using embeddings. We compare HDCT to two embedding-based retrieval models: RLM and SNRM. Results are shown in Table 8.5. As discussed in Section 8.4, we report their performance in the original papers, and re-evaluated HDCT using their methodology to be directly comparable [6].

As shown in Table 8.4, RLM was used in a pseudo-relevant back scenario (Retrieval+PRF). Its retrieval accuracy was lower than the other two methods. SNRM and HDCT can be used for stand-alone retrieval (Retrieval, where HDCT used BM25), or be combined with psuedo-relevance back (Retrieval+PRF, where HDCT used BM25+RM3). HDCT achieved higher absolute NDCG@20 values, and larger relative improvements from the corresponding *tf* baseline. In SNRM, documents are semantically matched by latent topics in embeddings, but may lose term specificity. This is a common issue in controlled vocabularies (Salton and McGill, 1984) and representation-based neural ranking models (Guo et al., 2016a). HDCT represents documents using the free-text vocabulary. It has higher precision as it preserves the exact term matching signals which are critical in information retrieval.

[6] NDCG@20 values of HDCT-inlink in Table 8.5 are different from Table 8.4 due to different evaluation methodologies. Zamani et al. (2018a); Zamani and Croft (2017) used the ClueWeb09-A qrels files. Table 8.5 followed the settings in (Dai et al., 2018; Dai and Callan, 2019b) and used ClueWeb09-B subset.

Table 8.6: Effectiveness of HDCT when trained with relevance labels and pseudo-relevance labels. Dataset: MS-MARCO-Doc. Superscripts 1-5 indicate statistically significant improvements over the corresponding methods, as labeled in the second column.

MS-MARCO-Doc		Dev Queries	
Retrieval Model	Indexing Term Weight	MRR	
BM25FE	1 *tf*	0.283	–
	2 HDCT-title	0.300^{13}	+6%
	3 HDCT-PRFaol	0.291^{1}	+3%
	4 HDCT-PRFmarco	0.307^{123}	+8%
	5 HDCT-supervised	0.320^{1234}	+13%

Summary. The analysis in this section shows that one can train effective HDCT models solely from the content of documents. HDCT is more accurate than strong *tf*-based baselines and state-of-the-art weakly-supervised embedding baselines. Being an efficient full-collection retrieval model, HDCT can be equally or more accurate than the supervised and more complex re-ranking models under low-resource or cold-start conditions.

8.5.2 Effects of Relevance and Pseudo-Relevance Labels

The previous experiment demonstrates the effectiveness of content-trained HDCT. The next experiment studies HDCT's performance when it was trained with stronger supervision from real search queries and relevance labels provided. Table 8.6 shows the effectiveness of HDCT models training using relevance labels and pseudo-relevance labels on the MS-MARCO-Doc dataset. We examined two types of pseudo-relevance labels that reflect what people might use in different settings: generated using in-domain and out-of-domain queries. All indexes were searched with BM25FE, the strongest retrieval model on MS-MARCO-Doc and also the most difficult case for HDCT.

HDCT-supervised is a fully-supervised model that used in-domain queries and true relevance labels. As shown in Table 8.6 , it is the most effective, outperforming the title-trained model by 7%, indicating that document titles are not necessarily aligned with user search intents.

HDCT-PRFaol used out-of-domain pseudo-relevance labels from AOL queries. Results in Table 8.6 show that they were not effective. We observed similar results on ClueWeb datasets. Domain difference is a common challenge in weak-supervision (MacAvaney et al., 2019b); this experiment reveals that it has a significant impact on HDCT.

HDCT-PRFmarco was trained on in-domain pseudo-relevance labels. It was the closest to the fully-supervised model, demonstrating that our PRF-based weak-supervision strategy is useful for in-domain queries. User activities, such as clicks, are not always accessible due to privacy regulations. In this case, the search queries alone can provide important evidence to build better document representations.

In summary, this experiment shows that the relevance-based supervision strategy can align HDCT with the search tasks, making it more effective than the content-trained models. Collecting relevance labels or user activities may be expensive or not permitted. Our PRF-based weak-supervision strategy provides a way to improve with the queries alone, which are often easier to collect.

Table 8.7: Effects of different ways of combining passages in HDCT. *ptf* stands for passage term frequency. DeepCT stands for the passage-level term weights from HDCT. HDCT was trained using document titles on MS-MARCO-Doc, and web inlinks on ClueWeb09-B. Retrieval model: BM25. Superscripts 1-9 indicate statistically significant improvements over the corresponding methods.

Method	MS-MARCO-Doc Dev Query MRR		ClueWeb09-B Title Query NDCG@20	
1 *tf*	0.245	–	0.272^{234}	–
2 DeepCT+TruncateDocument	0.265^{3}	+8%	0.200^{3}	-26%
3 DeepCT+PassageRetrievalAvg	0.243	-0%	0.148	-46%
4 DeepCT+PassageRetrievalMax	0.261^{3}	+7%	0.233^{23}	-14%
5 HDCT, sum	0.280^{1-4}	+14%	$\mathbf{0.303}^{1-4,6}$	+11%
6 HDCT, decay	$\mathbf{0.287}^{1-5}$	+ 17%	0.286^{1-4}	+5%

8.5.3 Effects of Hierarchical Document Modeling

HDCT uses a two-level hierarchy that first estimates term weights in passages using DeepCT, and then combine them into document-level term weights. This section first studies the effectiveness of HDCT's term weighting at the passage-level. It then compares different ways of combining passages. Finally, it analyzes its behavior through a case study.

Table 8.7 compare various alternative ways to combine passages. TruncateDocument truncates a document into a single passage of 512 tokens to directly fit DeepCT. PassageRetreival indexes and retrieves individual passages, and combines passage scores at the query time. A document's score is the average or maximum of its passage scores (Wu et al., 2019; Dai and Callan, 2019b; Kaszkiel and Zobel, 1997). Both methods were tested with our DeepCT-generated passage-level term weights (DeepCT).

As shown in Table 8.7, there is a large gap between DeepCT + TruncatedDocument and HDCT, indicating that the truncating a document is not sufficient and it is necessary to take into account all the words in documents.

PassageRetrieval makes use of all passages combining passage retrieval scores at the query time. As shown in Table 8.7, none of the passage retrieval approaches were as good as HDCT. They focus on the local content but lose the global context. For example, a passage discussing "the act to protect Yellowstone" is likely to be mistakenly retrieved by a query that looks for general "act" (Table 8.8, P6).

HDCT combines passage representations at the index time. Our results show that it is significantly better than truncating documents or combining passage scores. The effectiveness of passage weighting depend on the dataset. The simple unweighted sum is robust across dataset. The position-decayed sum (decay) is less effective on ClueWeb09-B, probably due to that the position decay is too strong for the longer documents in ClueWeb09-B.

Table 8.8 illustrates HDCT's hierarchical document modeling process. The Wikipedia web page of Yellowstone National Park from ClueWeb09-B contains over 10,000 words, and covers a wide range of topics including the park's history, geology, and recreation. As shown in Table 8.8, the original *tf* correctly favors important concepts like "yellowstone", but also favors non-content words such as "2007" and "retrieve".

HDCT successfully identifies essential terms in each passage, e.g., the act that created the park (P6), wildlife (P18), fires (P23), and scenes (P32). HDCT also recognized that these are different aspects of the

Table 8.8: An example of HDCT(-inlink) weighted passages of a ClueWeb09-B document. This document has 36 passages. P1-P36 show 5 terms with highest HDCT weights from 6 passages. The first and the last rows show 10 terms from the document with highest *tf* and HDCT weights.

Yellowstone National Park - Wikipedia		
tf	Doc	yellowstone: 247, park: 243, national: 147, 2007: 96, http:89, fire: 84, retrieve: 82, service:67, 03:47, year: 41
DeepCT	P1	yellow:10, stone:10, park:9, national:9, yellowstone:3
	P6	park:8, act:8, public: 3, 1872: 4, superintendent: 3
	P18	yellowstone:10, bison:6, herd:5, park:5, animal:4
	P23	fire:14, yellowstone:5, 1988:5, forest:8, rockies:4
	P32	yellowstone:10, volcano:7, lake:6, national:4, dome:2
	P36	wikipedia:1
HDCT	Doc (sum)	yellowstone:315, park:204, national:146, lake:31, grand:30, us:22, montana:22, fire:43, forest:33, teton:21

central topic "Yellowstone". These examples demonstrate that HDCT can identify passage-specific key terms as opposed to focusing on a single topic, thus it is applicable to documents with multiple topics or documents with different topics in different passages. In this work, the main purpose of breaking a document into passages is for efficiency. However, this case study suggests that even without efficiency constraints, it may still be beneficial to apply HDCT at a fine-grained discourse level to capture the local topics. We leave this for future research.

HDCT then sums up the passages, generating a document bag-of-words that characterizes the entire document. It correctly shows the central terms of the entire document, e.g. "yellowstone". It also provides an overview of the diverse topics discussed in different passages, such as 'fire', 'lake', and 'grand teton'. HDCT's document vector are more representative than *tf*, leading to higher retrieval effectiveness as shown in previous experiments.

Table 8.8 also shows how HDCT implicitly down-weights unimportant passages. P36 is the Wikipedia disclaimer "All text is available under the terms of...". Instead of giving high weights to locally-important terms, HDCT-inlink decides that the entire passage is not important. The highest term weight is 1 for 'wikipedia'; all other terms receive 0 weights. As a result, the disclaimer passage contributes little to the final document representation.

8.6 HDCT Summary

This chapter broadens the applications of DeepCT through three main directions. First, it presents HDCT, a hierarchical document term weighting framework, that extends the passage-level DeepCT for document retrieval. Second, it presents several weak-supervision strategies, allowing HDCT to be used in low-resource domains. Moreover, it studied the compatibility between our neural document term weights and widely-used pseudo-relevance feedback techniques.

In HDCT, a term's weight is an aggregation of its semantic importance in individual passages. The passage-level estimation uses deep, context-aware features from DeepCT. The output of HDCT is a document bag-of-words, allowing efficient and effective retrieval from an inverted index. In addition to typical retrieval models, the term weights are also compatible with a widely used pseudo-relevance feedback technique; moreover, it significantly *improves* pseudo-relevance feedback.

A content-based weak-supervision strategy is presented to train HDCT without depending on relevance labels. Experiments demonstrate its effectiveness. The content-trained HDCT achieved significant and robust improvements over standard *tf*-weighted retrieval models, strong embedding-based approaches, and supervised learning-to-rank systems. The content-based weak-supervision only relies on the document collection, making HDCT applicable to new collections and low-resource domains. Further study shows that search-specific labels such as queries and clicks can improve HDCT by aligning it with the user search intents. Our PRF-based weak-supervision strategy provides a way to leverage queries without using relevance labels or user clicks, which are sometimes harder to collect than queries.

Analysis demonstrates the advantages of HDCT's hierarchical document modeling approach. A passage retrieval experiment shows that HDCT better captures key terms in a passage than *tf*. The passage-level evidence cannot be directly combined by aggregating their retrieval scores. HDCT successfully translates its passages-level gains into document retrieval by combining passage term weights. HDCT provides the retrieval model with both global and passage-specific key terms, so that documents can be accurately retrieved through an efficient bag-of-word retrieval.

Part IV

Conclusion

Chapter 9

Conclusion

This chapter summarizes the work presented in the thesis, considers its major contributions, and discusses directions for future work.

9.1 Summary of Dissertation Results

For 50-60 years, information retrieval (IR) systems have relied on bag-of-words approaches. Bag-of-words is a shallow way of modeling natural language, but they are efficient and robust, outperforming many alternative retrieval models that use more complex NLP techniques that are slow and only cover limited patterns. However, the limitations in bag-of-words retrieval models remain unsolved. Recently, neural networks provide a powerful new tool to model natural language. This dissertation seeks to bring deeper language understanding into information retrieval using neural networks.

There are two fundamental limitations in most bag-of-words retrieval systems. First, to estimate the relevance between a query and a document, bag-of-words retrieval solely relies on the exact lexical match between the two pieces of text, disregarding the interactions among words that are lexically different. Second, in bag-of-words representations, a word's weight is usually estimated from its frequency, ignoring its meaning and its current linguistic context. This dissertation proposes a full range of neural network based solutions to address these challenges. Additionally, several techniques are developed to mitigate low-resource conditions, making our models generally applicable.

Existing learning-to-rank models heavily rely on lexical matching features. Distributed text representations allow matching every pair of words, generating much more evidence than exact lexical match, but the evidence is weaker and noisier. Many neural ranking approaches tried to soft match words using distributed representations such as word2vec (Mikolov et al., 2013) and GloVE (Pennington et al., 2014), but they only have a limited history of success for document ranking. This dissertation starts by proposing `K-NRM` (Chapter 3), a neural ranking model with a novel kernel-pooling technique that can group these soft match signals and distinguish useful signals from noisy ones. It uses multiple Gaussian kernels to count soft matches at different similarity levels. When trained by relevance feedback data, the kernels organize soft match patterns into groups based on how they contribute to the relevance. Experiments demonstrated that K-NRM outperforms strong feature-based learning-to-rank models that defined the state-of-the-art at the time, demonstrating the power of neural ranking models for the first time.

Besides matching individual words, the query and document often match at n-grams. How to effectively model n-gram soft-matches remains an open question in neural IR. Treating n-grams as discrete words will explode the parameter space and cause data sparsity issues. Chapter 4 of this dissertation proposes `Conv-KNRM`, which can generate n-gram on the fly by composing adjacent words' embeddings

to n-gram embeddings using convolutional neural networks. Conv-KNRM then matches the n-grams in the embedding space and uses K-NRM's kernel-pooling technique to learn different groups of n-gram soft match signals. With such an architecture, Conv-KNRM discovers IR-customized phrase soft match patterns, such as "farm" to "eat & drink". Experiments show that Conv-KNRM further improved K-NRM, which was the previous state-of-the-art, by up to 30% on two datasets from different languages. Our feature weight analysis shows that in Conv-KNRM, 78% of the feature weights go to n-gram match while only 22% go to unigram match, and over 80% of the weights are assigned to soft match signals, showing the importance of soft matching n-grams in search ranking.

People search for a wide range of data every day, for example, articles, images, and products. Traditionally, they require very different data representations, feature extractors, and corresponding retrieval systems. Although those retrieval systems may face several common issues, e.g., how to model soft match, the distinct data representations and features make it prohibitive to re-use a retrieval system on a different media. Neural networks, with the use of distributed representations and the ability to learn from scratch, make it possible to use similar architectures for a variety of search tasks without designing individual data representations and features. Chapter 5 of this dissertation investigates whether the model architectures we developed for text can be re-used on other data modalities by testing them on the engineering diagram search task. It modifies Conv-KNRM to accept image local embeddings and to use 2-D convolutional neural networks to capture spatial relations. The modified model achieved strong performance on two engineering diagram retrieval tasks, indicating that the proposed ranking models do address the general search issues, and can be shared across different data modalities.

K-NRM and Conv-KNRM, along with their concurrent neural ranking models, focus on learning query-document relevance patterns from search-specific training data. However, such approaches require large amounts of training data, which are expensive to collect, prohibiting our neural ranking models from being used in low-resource search domains. On the other hand, a large portion of the knowledge can be found from the documents without observing search queries or clicks. In Chapter 6, this dissertation research investigated leveraging pre-trained deep language models to improve ranking in low-resource search domains where training data are limited, and studies how the pre-trained language knowledge impacts neural ranking models. We develop the DocBERT reranker that uses a deep pre-trained language model BERT (Lee et al., 2019a) to rank documents. To address the input length limitation of BERT, DocBERT uses a passage-retrieval framework and a distant supervision strategy to train the passage-level model on the document-level labels. DocBERT shows competitive accuracy on two low-resource datasets, while all other neural ranking models fail to outperform a feature-based learning-to-rank model due to lack of training data. A more important finding is that the DocBERT reranker brings substantial improvements to natural language queries by leveraging the sentence structures. While the research discussed previously mostly focuses on modeling the query-document interactions, DocBERT shows potential in modeling the query/document content.

Inspired by DocBERT's language modeling ability, this dissertation shifts its focus from document-query interaction to document/query representation, with the goal to build deep but efficient text representations for the initial retrieval stage. Most initial rankers are older-but-efficient bag-of-words retrieval models that use term frequency signals. However, frequency-based term weighting does not necessarily reflect a term's importance in queries and documents, especially when the frequency distribution is flat, such as in sentence-long queries or passage-long documents. Chapter 7 proposes DeepCT, a new term weighting approach that explicitly uses BERT to estimate term importance based on its current frame of context. When applied to documents, it generates *tf*-like term weights that can be stored in a typical inverted index for efficient bag-of-words retrieval. Experimental results show that an index built with DeepCT weights improves the accuracy of two popular retrieval algorithms by up to 50%. Running

	Retrieval	Reranking	MRR@10
Dissertation PART II	BM25	Learning-To-Rank* (Official Baseline)	0.195
		K-NRM* (Chap. 3)	0.218
		Conv-KNRM* (Chap. 4)	0.247
		Conv-KNRM ensemble* (Chap. 3)	0.290
		DocBERT Reranker (Chap. 6)	0.364
Dissertation PART III	BM25* (Official Baseline)	None	0.167
	DeepCT (Chap. 7)		0.243
	DeepCT+BM25FE (Chap. 8)		0.250
Combined	DeepCT (Chap. 7)	Conv-KNRM (Chap. 4)	0.278
	DeepCT (Chap. 7)	BERT Reranker (Nogueira and Cho, 2019)	0.376
	DeepCT+BM25FE (Chap. 8)	DocBERT Reranker (Chap. 6)	0.394
	DeepCT (Chap. 7)	TF Ranking (Najork et al., 2020)	0.405

Table 9.1: Results of the algorithms developed in this thesis the on MS MARCO Passage Ranking dataset dev set. * denotes results reported by other researchers on the MS MARCO official website[1].

BM25 on the index can be as effective as several previous state-of-the-art multi-stage search systems that use knowledge bases, machine learning, and large amounts of training data. Analysis shows compared to classic *tf* term weights, DeepCT can find the most central words in a text even if they are mentioned only once. Non-central words, even if mentioned frequently in the text, are suppressed. Such behavior is uncommon in previous term weighting approaches. We view DeepCT as an encouraging step from "frequencies" to "meanings".

Although showing promising results, DeepCT has several limitations. First, it is limited by BERT's input length constraints, therefore it only supports short text. Second, it requires relevant query-document pairs for training, which are often not available. Chapter 8 addresses the first issue with a hierarchical document term weighting framework, called HDCT. In HDCT, a term's weight is an aggregation of its semantic importance in individual passages. The passage-level estimation uses deep, context-aware features from DeepCT. The output is a document bag-of-words that allows efficient and effective retrieval from an inverted index. Chapter 8 also proposes two weak supervision strategies for HDCT, allowing it to be used in low resource scenarios. Experiments show that trained solely on the web anchor text, HDCT can achieve significant and robust improvements over standard *tf*-weighted retrieval models, strong embedding-based approaches, and supervised learning-to-rank systems. Analysis shows that HDCT can emphasize the global document topical terms, identify the local passage-specific key terms, and suppress noisy terms and passage, leading to accurate and efficient bag-of-word retrieval. HDCT's deep context-aware bag-of-words representations not only support standard BM25 retrieval, but are also compatible with pseudo-relevance feedback retrieval models, indicating many new possibilities of improving existing BoW retrieval methods using this new bag-of-words representation.

Table 9.1 illustrates how this dissertation gradually advances today's IR systems with deeper language understanding. It evaluates some of our main algorithms on the MS MARCO Passage Ranking dataset. At the beginning of this dissertation research, information retrieval systems were using bag-of-words retrieval (BM25) for initial retrieval and features-based learning-to-rank for reranking. This dissertation advances both stages. On the reranking side, K-NRM effectively models soft match patterns among queries and documents, successfully surpassing the learning-to-rank baseline. The n-gram soft matching (Conv-KNRM) and ensemble techniques developed in this dissertation further improved the reranking accuracy. On the initial retrieval side, DeepCT replaces the long-used *tf* term weights with its context-aware term weights, generating new state-of-the-art for the initial retrieval stage. More importantly, the rankers and rerankers

can be used as building blocks to generate more powerful retrieval systems. We believe this thesis has made a solid step towards the next-generation retrieval systems that are equipped with deep understanding of both general language knowledge and search-specific language usages.

9.2 Thesis Contribution

This dissertation develops a suite of tools to improve language understanding in information retrieval. Besides the specific algorithms, we believe this dissertation introduces important changes to how people will view and build text retrieval systems.

This dissertation research gives a new understanding of the language uses in search – that general language patterns obtained from task-neutral corpus analysis do provide values, but they can be quite different from how people use languages when searching information. Previous research has devoted considerable efforts to using semantically similar words to soft match queries and documents. Most of the previous research used linguistically motivated word similarity measurements as derived from the corpus, assuming that words with similar linguistic usages will also indicate strong relevance in search. This dissertation research shows that this assumption is often not true. Our analysis found that more than 90% of word pairs that are mapped together in word2vec are decoupled by `K-NRM`, while word pairs that are less correlated in documents but convey frequent search tasks are discovered and mapped to certain similarity levels. The newly discovered soft match pairs require search-specific knowledge, such as search intents (e.g., "farm" and "eat & drink") and site relationships (e.g., "Friends" and "NetFlix"), and cannot be captured by the common linguistic properties defined in NLP research. Similar observations were found on stronger general-purpose language models like BERT, where although the pre-trained model can perform reasonably well when trained with a few search examples, to get real improvements, the models still need to learn search-specific language knowledge from a large amount of search data.

After identifying the importance of search-specific knowledge, this dissertation research revisits, replicates and extends research on creating search-specific training data. It studies a wide range of techniques to reduce the number of search logs needed to train IR models, including borrowing search logs from related domains, generating pseudo-labels using a weaker search engine, and mining query-like signals from the document content. It confirms prior work showing that when search data is unavailable, titles, inlinks, and other brief summaries of a document are good surrogates for relevance data. This dissertation also shows that document-level relevance signals can be converted to useful passage-level and token-level signals, which has not been done much in the past.

This dissertation broadens the scope of neural IR research. It shows that neural models have two distinct contributions to more accurate retrieval: text understanding, and matching. Importantly, while the matching needs to be done online, the text understanding part can be done offline. Before our work, little attention was paid to the text understanding aspect of neural IR research. Most prior work focused on matching, using neural networks as a black-box to compute query-document relevance scores. As query-document interactions must be modeled at query time, they raise many concerns about efficiency. Also, the focus on query-document matching limits the generalization ability of the models to rare or unseen queries. This dissertation research was one of the first to use a BERT reranker and to identify its advantages in document understanding – in addition to modeling query-document interactions, BERT excels at capturing the language structures of queries and documents. `DeepCT` and `HDCT` took a step further and explicitly use deep neural network models to analyze the contents of documents, pushing neural IR research to go beyond matching. The landscape this thesis research sets up provides a wide range of opportunities for future research in neural IR.

This dissertation research changes how people will build indexes and retrieval pipelines. During the past decade, efforts to improve ranking were mostly focused on the rerankers, which evolved from shallow feature-based models to complex deep neural networks. Meanwhile, indexing and initial retrieval stages in IR have not changed much. Although it was always believed that retrieval models based on bag-of-words and frequency signals have many limitations, prior approaches that use parsing or graph algorithms mostly failed as they are difficult to train and slow to use at a large scale. This dissertation provides a new way to assign index terms to documents by offline running a deep language model. They show that term frequency is no longer sufficient, and that substantial improvements can be made in the initial retrieval stage with the new text understanding tools. The improved initial retrieval stage has additive effects in the end-to-end retrieval pipeline, which is uncommon in prior research. These results indicate that it is time to revisit our indexing systems and retrieval models, and to move from shallow heuristics towards deep document understanding. In fact, several of the top MS MARCO competition systems use `DeepCT` in replacement of the long-standing BM25 baseline, demonstrating that this dissertation research has already made an impact on the field. We believe our research will be a beginning of many changes and innovations in the field of information retrieval.

9.3 Future Directions

The dissertation identifies several advantages of using neural networks for information retrieval, opening new directions for research far beyond their current use. We now discuss future work in three directions, which have large potential and are currently not well understood.

One direction is to improve the initial retrieval stage with deep text understanding. The initial retrieval stage requires simple data representations and simple algorithms for efficiency. The context-aware bag-of-words document representations built in this dissertation show that it is possible to improve the quality of text representation while retaining the simple forms. It opens up several research opportunities for further improvements and investigations.

`DeepCT` and `HDCT` provide a framework to estimate the probability of assigning an index term to a specific document based on a deep content understanding of the document. Our current models only consider individual words in a document. A possible next step is to incorporate other indexable text units such as n-gram and phrases. Moreover, the methods developed in this dissertation focus on reweighing existing words. They help little when the document and the query have a vocabulary mismatch. A possible improvement is document expansion – to expand the candidate index term set to the entire vocabulary, letting our models select additional index terms that are close to the meaning of the document. This approach has the potential to increase recall in initial retrieval. More interestingly, the vocabulary is not limited to the current document collection. It can also be words frequently used in the queries, words from a high-quality controlled vocabulary, or words from other languages.

More broadly, the idea is to decouple complex neural IR models into "understanding + matching". The matching must be done online. The text understanding can be done offline, generating text representations that can accelerate evaluation. This idea is not limited to our current methods that use bag-of-words. Instead, a document could also be represented as a latent embedding learned from a deep neural network, a set of intermediate results extracted from a neural ranker, or a combination of several types of representations. We are encouraged to think about which types of representations best support effective a search. Taking a step further, with the methods developed in this dissertation, indexing is becoming a machine learning process. It may be possible to train the entire retrieval stack together, including indexing, initial retrieval, and reranking. In such an end-to-end retrieval system, feedback from the end users can be back-propagated through the entire stack, so that the earlier stage rankers can adjust themselves to better

support later stages, and the index can fix errors in the document representations. The pipelined retrieval framework has been used for decades. This dissertation shows how better text understanding is leading us to new, data-driven deep retrieval systems. We believe this will lead to more Ph.D. dissertations in the near future.

A second future direction is to leverage document understanding techniques to develop new ways of result presentation in search engines, advancing how search engines interact with humans. Our research has focused on document retrieval and ranking. After the documents are retrieved, they are usually displayed as a plain list. Such ways of result presentation are not optimal, especially for small-screen devices or speech-based systems, which are becoming increasingly popular. With better text understanding tools, it is possible for search engines to read the results, understand each individual document and the relations among them, and present the information to people in more efficient ways. For example, the context-aware term weighting techniques developed in this dissertation can extract aspects and topics from text documents. A voice-based search engine can use these techniques to give the user a concise summary of the main aspects covered by the result set, so that the user can ask follow-up questions for more specific information. Future search engines will be able to proactively talk to people and assist people seamlessly instead of reacting to user's search queries and clicks. A key factor towards that goal is to understand the search results. This dissertation provides some initial tools for this goal; we hope more will come in the future.

A third direction is to develop systematic approaches to building neural IR models in low-resource domains, generalizing our solutions to enterprise search. This dissertation mostly focuses on web search, where the documents are from the open web, the search engine has high search traffic, and it is relatively easy to get sufficient training data. Enterprise search, on the other hand, searches information within an enterprise, so the documents are often domain-specific, there is possibly no existing traffic, and it is likely to face cold-start problems. As various parts of this dissertation studied the generalization ability of neural IR models, we believe an important next step is to replicate and extend these studies on enterprise search. This dissertation developed various unsupervised signals to mimic search data using titles, inlinks, and pseudo-relevance labels. However, the unsupervised signals depend on the specific formats of the data, people's prior knowledge about the domain, and many manual trial-and-error processes. A next step would be to develop a systematic and automatic way to find such unsupervised signals, for example, using meta-learning methods that can automatically select the most effective data for a specific domain. Another promising direction is IR-specific pre-training. This dissertation uses pre-trained general-purpose language models for fine-tuning, but there is a discrepancy between the pre-training tasks and the IR task. Pre-trained on slot-filling tasks, language models like BERT may fail to capture some IR patterns (e.g., exact match and term frequencies), and may not be sufficient to model domain-specific knowledge (e.g., medical and legal knowledge). Future research should investigate whether search-specific knowledge can be learned during pre-training, what are the suitable IR signals for pre-training, and how IR-specific pre-training differs in specific domains. Enterprise search is a billion-dollar market, but existing public search engines are still old bag-of-words systems, and there is no easy way for an enterprise to try out and get the benefits from deep learning. We believe this dissertation research provides encouraging evidence and a pool of algorithms for future research towards reshaping enterprise search with advanced language technologies.

Bibliography

Qingyao Ai, Liu Yang, Jiafeng Guo, and W. Bruce Croft. Analysis of the paragraph vector model for information retrieval. In *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval*, pages 133–142, 2016.

James Allan and Hema Raghavan. Using part-of-speech patterns to reduce query ambiguity. In *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 307–314, 2002.

Nima Asadi, Donald Metzler, Tamer Elsayed, and Jimmy J. Lin. Pseudo test collections for learning web search ranking functions. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1073–1082, 2011.

Martin Aumüller, Tobias Christiani, Rasmus Pagh, and Francesco Silvestri. Distance-sensitive hashing. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 89–104, 2018.

Artem Babenko and Victor S. Lempitsky. Aggregating local deep features for image retrieval. In *2015 IEEE International Conference on Computer Vision*, pages 1269–1277, 2015.

Artem Babenko, Anton Slesarev, Alexander Chigorin, and Victor S. Lempitsky. Neural codes for image retrieval. In *Computer Vision - ECCV 2014 - 13th European Conference, Proceedings, Part I*, volume 8689 of *Lecture Notes in Computer Science*, pages 584–599, 2014.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 238–247, 2014.

Richard K. Belew. *Adaptive Information Retrieval: Machine Learning in Associate networks*. PhD thesis, Michigan University, Ann Arbor (USA), 1986.

Richard K. Belew. Adaptive information retrieval: Using a connectionist representation to retrieve and learn about documents. In *Proceedings of the 12th International Conference on Research and Development in Information Retrieval*, pages 11–20, 1989.

Michael Bendersky, Donald Metzler, and W. Bruce Croft. Learning concept importance using a weighted dependence model. In *Proceedings of the Third International Conference on Web Search and Web Data Mining*, pages 31–40, 2010.

Michael Bendersky, Donald Metzler, and W. Bruce Croft. Parameterized concept weighting in verbose queries. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 605–614, 2011.

Michael Bendersky, Donald Metzler, and W. Bruce Croft. Effective query formulation with multiple information sources. In *Proceedings of the Fifth International Conference on Web Search and Web*

Data Mining, pages 443–452, 2012.

Richard Berendsen, Manos Tsagkias, Wouter Weerkamp, and Maarten de Rijke. Pseudo test collections for training and tuning microblog rankers. In *The 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 53–62, 2013.

Adam L. Berger and John D. Lafferty. Information retrieval as statistical translation. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 222–229, 1999.

Roi Blanco and Christina Lioma. Graph-based term weighting for information retrieval. *Information Retrieval*, 15(1):54–92, 2012.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. In *Advances in Neural Information Processing Systems 14*, pages 601–608. MIT Press, 2001.

M. Boughanem and Chantal Soulé-Dupuis. Query expansion and neural network. In *Proceedings of 4th International Conference of Computer-Assisted Information Retrieval*, pages 519–533, 1994.

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a Siamese time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1993.

Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, pages 89–96, 2005.

William R. Caid, Susan T. Dumais, and Stephen I. Gallant. Learned vector-space models for document retrieval. *Information Processing & Management*, 31(3):419–429, 1995.

James P. Callan. Passage-level evidence in document retrieval. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 302–310, 1994.

Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. Selecting good expansion terms for pseudo-relevance feedback. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 243–250, 2008.

Matteo Catena, Ophir Frieder, Cristina Ioana Muntean, Franco Maria Nardini, Raffaele Perego, and Nicola Tonellotto. Enhanced news retrieval: Passages lead the way! In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1269–1272, 2019.

Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. In *Proceedings of the Yahoo! Learning to Rank Challenge*, pages 1–24, 2011.

Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. *Click Models for Web Search*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2015.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. What does BERT look at? an analysis of BERT's attention. *arXiv preprint arXiv:1906.04341*, 2019.

Alexis Conneau and Guillaume Lample. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*, pages 7057–7067, 2019.

William S. Cooper, Fredric C. Gey, and Daniel P. Dabney. Probabilistic retrieval based on staged logistic regression. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and*

Development in Information Retrieval, pages 198–210, 1992.

Gordon V. Cormack, Mark D. Smucker, and Charles L. A. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval*, 14(5):441–465, 2011.

Nick Craswell, Onno Zoeter, Michael J. Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the International Conference on Web Search and Web Data Mining*, pages 87–94, 2008.

Nick Craswell, W. Bruce Croft, Jiafeng Guo, Bhaskar Mitra, and Maarten de Rijke. Report on the SIGIR 2016 Workshop on Neural Information Retrieval (Neu-IR). In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 96–103, 2016.

W. Bruce Croft and David J. Harper. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation*, 35(4):285–295, 1979.

W. Bruce Croft, Donald Metzler, and Trevor Strohman. *Search Engines - Information Retrieval in Practice*. Pearson Education, 2009.

Balázs Csanád Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

Zhuyun Dai and Jamie Callan. Context-aware sentence/passage term importance estimation for first stage retrieval. *arXiv preprint arXiv:1910.10687*, 2019a.

Zhuyun Dai and Jamie Callan. Deeper text understanding for IR with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 985–988, 2019b.

Zhuyun Dai and Jamie Callan. Context-aware term weighting for first-stage passage retrieval. In *To appear in the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020a.

Zhuyun Dai and Jamie Callan. Context-aware document term weighting for ad-hoc search. In *Proceedings of The Web Conference 2020*, pages 1897–1907, 2020b.

Zhuyun Dai, Chenyan Xiong, and Jamie Callan. An evaluation of the kernel based neural ranking model in ntcir13 www. In *Proceedings of the 13th NTCIR Conference on Evaluation of Information Access Technologies*, 2017.

Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, pages 126–134, 2018.

Zhuyun Dai, Zhen Fan, Hafeezul Rahman, and Jamie Callan. Local matching networks for engineering diagram search. In *The World Wide Web Conference*, pages 340–350, 2019.

Jeffrey Dalton, Laura Dietz, and James Allan. Entity query feature expansion using knowledge base links. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 365–374, 2014.

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 65–74, 2017.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

Fernando Diaz, Bhaskar Mitra, and Nick Craswell. Query expansion with locally-trained word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

Laura Dietz, Manisha Verma, Filip Radlinski, and Nick Craswell. Trec complex answer retrieval overview. In *Proceedings of the Twenty-Seventh Text REtrieval Conference*, 2017.

Nadav Eiron and Kevin S. McCurley. Analysis of anchor text for web search. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 459–460, 2003.

Mathias Eitz, Kristian Hildebrand, Tamy Boubekeur, and Marc Alexa. Sketch-based image retrieval: Benchmark and bag-of-features descriptors. *IEEE Transactions on Visualization and Computer Graphics*, 17(11):1624–1636, 2011.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. Modeling diverse relevance patterns in ad-hoc retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 375–384, 2018.

Manuel J. Fonseca, Alfredo Ferreira, and Joaquim A. Jorge. Content-based retrieval of technical drawings. *International Journal of Computer Applications in Technology*, 23(2-4):86–100, 2005.

Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 170–178, 1998.

Norbert Fuhr and Chris Buckley. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems (TOIS)*, 9(3):223–248, 1991.

George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30(11):964–971, 1987.

Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth J. F. Jones. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 795–798, 2015.

Jianfeng Gao, Xiaodong He, and Jian-Yun Nie. Clickthrough-based translation models for web search: From word models to phrase models. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management*, pages 1139–1148, 2010.

Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *Proceedings of Computer Vision - ECCV 2014 - 13th European Conference*, volume 8695 of *Lecture Notes in Computer Science*, pages 392–407, 2014.

Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *10th IEEE International Conference on Computer Vision*, pages 1458–1465, 2005.

Warren R. Greiff. A theory of term weighting based on exploratory data analysis. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 11–19, 1998.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, pages 55–64, 2016a.

Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. Semantic matching by non-linear word transportation for information retrieval. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, pages 701–710, 2016b.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*, 2020.

Christophe Van Gysel, Maarten De Rijke, and Evangelos Kanoulas. Neural vector spaces for unsupervised information retrieval. *ACM Transactions on Information Systems (TOIS)*, 36(4):38:1–38:25, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Proceedings of Computer Vision - ECCV 2014 - 13th European Conference*.

Sebastian Hofstätter, Navid Rekabsaz, Carsten Eickhoff, and Allan Hanbury. On the effect of low-frequency terms on neural-ir models. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1137–1140, 2019.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in Neural Information Processing Systems 27*, pages 2042–2050, 2014.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry P. Heck. Learning deep structured semantic models for web search using clickthrough data. In *22nd ACM International Conference on Information and Knowledge Management*, pages 2333–2338, 2013.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. PACRR: A position-aware neural IR model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058, 2017.

Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. Co-pacrr: A context-aware neural IR model for ad-hoc retrieval. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 279–287, 2018.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. TinyBERT: Distilling BERT for natural language understanding. *CoRR*, abs/1909.10351, 2019.

Albert Jimenez, Jose M. Alvarez, and Xavier Giró i Nieto. Class weighted convolutional features for visual instance search. In *Proceedings of the British Machine Vision Conference 2017*, 2017.

Rong Jin, Alexander G. Hauptmann, and ChengXiang Zhai. Title language model for information retrieval. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–48, 2002.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142, 2002.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 60(5):493–502, 1972.

Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *Proceedings of the 15th European Conference on Computer Vision*, pages 685–701, 2016.

Maryam Karimzadehgan and ChengXiang Zhai. Estimation of statistical translation models based on mutual information for ad hoc information retrieval. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 323–330, 2010.

Marcin Kaszkiel and Justin Zobel. Passage retrieval revisited. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185, 1997.

Marcin Kaszkiel and Justin Zobel. Effective ranking with arbitrary passages. *Journal of the American Society for Information Science and Technology*, 52(4):344–364, 2001.

Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751, 2014.

Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. Boilerplate detection using shallow text features. In *Proceedings of the Third International Conference on Web Search and Web Data Mining*, pages 441–450, 2010.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114, 2012.

Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems 7*, pages 231–238, 1994.

K. L. Kwok. A neural network for probabilistic information retrieval. In *Proceedings of the 12th International Conference on Research and Development in Information Retrieval*, pages 21–30, 1989.

John Lafferty and Chengxiang Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 111–119, 2001.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Victor Lavrenko and W. Bruce Croft. Relevance-based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 6086–6096, 2019a.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. *arXiv preprint arXiv:1906.00300*, 2019b.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems 27*, pages 2177–2185, 2014.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *arXiv preprint arXiv:1908.03557*, 2019.

Jimmy Lin. The neural hype and comparisons against weak baselines. In *ACM SIGIR Forum*, volume 52, pages 40–51, 2019.

Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 115–124, 2017a.

Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, pages 225–331, 2009.

Xiaoyong Liu and W Bruce Croft. Passage retrieval based on language models. In *Proceedings of the 11th International Conference on Information and Knowledge Management*. ACM, 2002.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Yiqun Liu, Xiaohui Xie, Chao Wang, Jian-Yun Nie, Min Zhang, and Shaoping Ma. Time-aware click model. *ACM Transactions on Information Systems*, pages 16:1–16:24, 2017b.

Zhiyong Lu, Won Kim, and W. John Wilbur. Evaluation of query expansion using MeSH in PubMed. *Information Retrieval Journal*, pages 69–80, 2009.

Cheng Luo, Tetsuya Sakai, Yiqun Liu, Zhicheng Dou, Chenyan Xiong, and Jingfang Xu. Overview of the NTCIR-13 We Want Web task. *Proceedings of NTCIR-13*, 2017a.

Cheng Luo, Yukun Zheng, Yiqun Liu, Xiaochuan Wang, Jingfang Xu, Min Zhang, and Shaoping Ma. SogouT-16: A new web corpus to embrace IR research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1233–1236, 2017b.

Mathias Lux and Oge Marques. *Visual Information Retrieval Using Java and LIRE*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2013.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. Cedr: Contextualized embeddings for document ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1101–1104, 2019a.

Sean MacAvaney, Andrew Yates, Kai Hui, and Ophir Frieder. Content-based weak supervision for ad-hoc re-ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 993–996, 2019b.

Joel Mackenzie, Zhuyun Dai, Luke Gallagher, and Jamie Callan. Efficiency implications of term re-weighting for passage retrieval. In *To appear in the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

Donald Metzler and W. Bruce Croft. A markov random field model for term dependencies. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 472–479, 2005.

Donald Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Information Retrieval*, pages 257–274, 2007.

Douglas P Metzler and Stephanie W Haas. The constituent object parser: syntactic structure matching for information retrieval. *ACM Transactions on Information Systems (TOIS)*, 7(3):292–316, 1989.

Rada Mihalcea and Paul Tarau. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, 2004.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119, 2013.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*, 2017.

Bhaskar Mitra and Nick Craswell. An updated duet model for passage re-ranking. *arXiv preprint arXiv:1903.07666*, 2019.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1291–1299, 2017.

Bhaskar Mitra, Corby Rosset, David Hawking, Nick Craswell, Fernando Diaz, and Emine Yilmaz. Incorporating query term independence assumption for efficient retrieval and ranking using deep neural networks. *arXiv preprint arXiv:1907.03693*, 2019.

Stefan Müller and Gerhard Rigoll. Engineering drawing database retrieval using statistical pattern spotting techniques. In *Proceedings of the 3rd International Workshop on Graphics Recognition Recent Advances*, pages 246–255, 1999.

Marc Najork, Mike Bendersky, Shuguang Han, and Xuanhui Wang. Learning-to-rank with bert in tf-ranking. Technical report, Google Research, 2020.

Eric T. Nalisnick, Bhaskar Mitra, Nick Craswell, and Rich Caruana. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference on World Wide Web*, pages 83–84, 2016.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Nils Nilsson. *Learning Machines: Foundations of Trainable Pattern-Classifying Systems*. 1965.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *arXiv:1901.04085*, 2019.

Rodrigo Nogueira, Kyunghyun Cho, Yang Wei, Lin Jimmy, and Kyunghyun Cho. Document expansion by query prediction. *arXiv:1904.08375*, 2019.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Kaiyue Pang, Yi-Zhe Song, Tony Xiang, and Timothy M. Hospedales. Cross-domain generative learning for fine-grained sketch-based image retrieval. In *Proceedings of the British Machine Vision Conference*, 2017a.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. A study of MatchPyramid models on ad-hoc retrieval. *arXiv preprint arXiv:1606.04648*, 2016a.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. Text matching as image recognition. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages

2793–2799, 2016b.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 257–266, 2017b.

Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A picture of search. In *Proceedings of the 1st international conference on Scalable information systems*, pages 1–es, 2006.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2018.

Jay Michael Ponte and W Bruce Croft. *A language modeling approach to information retrieval*. PhD thesis, University of Massachusetts at Amherst, 1998.

Mary Arpita Pyreddy, Varshini Ramaseshan, Narendra Nath Joshi, Zhuyun Dai, Chenyan Xiong, Jamie Callan, and Zhiyuan Liu. Consistency and variation in kernel neural ranking model. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 961–964, 2018.

Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. In *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval*, 2007.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

Fiana Raiber and Oren Kurland. Query-performance prediction: setting the expectations straight. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 13–22, 2014.

T. B. Rajashekar and W. Bruce Croft. Combining automatic and manual index representations in probabilistic retrieval. *Journal of the American society for Information science*, pages 272–283, 1995.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

Navid Rekabsaz, Mihai Lupu, Allan Hanbury, and Hamed Zamani. Word embedding causes topic shifting; Exploit global context! In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1105–1108, 2017.

Stephen E Robertson and Steve Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, 1994.

François Rousseau and Michalis Vazirgiannis. Graph-of-word and tw-idf: new approach to ad hoc ir. In *22nd ACM International Conference on Information and Knowledge Management*, 2013.

Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. Using word embeddings for automatic query expansion. *arXiv preprint arXiv:1606.07608*, 2016.

Ruslan Salakhutdinov and Geoffrey E. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, pages 969–978, 2009.

Gerard Salton and Michael McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1984.

Gerard Salton, James Allan, and Chris Buckley. Approaches to passage retrieval in full text information systems. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, 1993.

Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. Evaluation methods for unsupervised word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015.

Hinrich Schütze. Word space. In *Advances in neural information processing systems*, pages 895–902, 1993.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 101–110, 2014a.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International World Wide Web Conference*, pages 373–374, 2014b.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the 9th IEEE International Conference on Computer Vision*, pages 1470–1477, 2003.

Aya Soffer, David Carmel, Doron Cohen, Ronald Fagin, Eitan Farchi, Michael Herscovici, and Yoëlle S. Maarek. Static index pruning for information retrieval systems. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–50, 2001.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mass: Masked sequence to sequence pre-training for language generation. *arXiv preprint arXiv:1905.02450*, 2019.

Trevor Strohman, Donald Metzler, Howard Turtle, and W Bruce Croft. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, pages 2–6, 2005.

Tomek Strzalkowski. Natural language information retrieval. *Information Processing & Management*, 31 (3):397–417, 1995.

Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7464–7473, 2019a.

Siqi Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for BERT model compression. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019*

Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 4322–4331, 2019b.

Krysta M Svore and Christopher JC Burges. A machine learning approach for improved BM25 retrieval. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 1811–1814, 2009.

Tao Tao and ChengXiang Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 162–169, 2006.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 4593–4601, 2019a.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. What do you learn from context? probing for sentence structure in contextualized word representations. In *7th International Conference on Learning Representations*, 2019b.

Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of CNN activations. *arXiv Preprint arXiv:1511.05879*, 2015.

Ilya Trofimov, Anna Kornetova, and Valery Topinskiy. Using boosted trees for click-through rate prediction for sponsored search. In *Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy*, page 2, 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 6000–6010, 2017.

Visual Geometry Group, Department of Engineering Science, University of Oxford. Very deep convolutional networks for large-scale visual recognition, 2018. `http://www.robots.ox.ac.uk/~vgg/research/very_deep/`, Last accessed on 2018-10-15.

Mengqiu Wang and Luo Si. Discriminative probabilistic models for passage based retrieval. In Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong, editors, *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 419–426, 2008.

Xing Wei and W. Bruce Croft. LDA-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 178–185, 2006.

S. K. Michael Wong, Y. J. Cai, and Yiyu Yao. Computation of term associations by a neural network. In *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 107–115, 1993.

Qiang Wu, Christopher J. C. Burges, Krysta Marie Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 2010.

Zhijing Wu, Jiaxin Mao, Yiqun Liu, Min Zhang, and Shaoping Ma. Investigating passage-level relevance and its role in document-level relevance judgment. In *Proceedings of the 42nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019.

Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25 th International Conference on Machine Learning*,

pages 1192–1199, 2008.

Chenyan Xiong and Jamie Callan. Esdrank: Connecting query and documents through external semi-structured data. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pages 951–960, 2015.

Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. Bag-of-entities representation for ranking. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, pages 181–184. ACM, 2016.

Chenyan Xiong, Jamie Callan, and Tie-Yan Liu. Word-entity duet representations for document ranking. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 763–772, 2017a.

Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 55–64, 2017b.

Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Enabling the use of lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1253–1256. ACM, 2017.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019*, pages 5754–5764, 2019.

Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly, Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, et al. Ranking relevance in yahoo search. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 323–332, 2016.

Hamed Zamani and W. Bruce Croft. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 505–514. ACM, 2017.

Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik G. Learned-Miller, and Jaap Kamps. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 497–506, 2018a.

Hamed Zamani, Bhaskar Mitra, Xia Song, Nick Craswell, and Saurabh Tiwary. Neural ranking models with multiple document fields. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, pages 700–708, 2018b.

Hugo Zaragoza, Nick Craswell, Michael J Taylor, Suchi Saria, and Stephen E Robertson. Microsoft Cambridge at TREC 13: Web and hard tracks. In *Proceedings of the Twenty-Seventh Text REtrieval Conference*, 2004.

Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS)*, 22(2):179–214, 2004.

Huaping Zhang, Hongkui Yu, Deyi Xiong, and Qun Liu. HHMM-based Chinese lexical analyzer ICTCLAS. In *Proceedings of the Second Workshop on Chinese Language Processing*, 2003.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. ERNIE: enhanced language representation with informative entities. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 1441–1451, 2019.

Le Zhao. *Modeling and solving term mismatch for full-text retrieval*. PhD thesis, Carnegie Mellon University, 2012.

Le Zhao and Jamie Callan. Term necessity prediction. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management*, pages 259–268, 2010.

Guoqing Zheng and Jamie Callan. Learning to reweight terms with distributed representations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 575–584, 2015.

Guido Zuccon, Bevan Koopman, Peter Bruza, and Leif Azzopardi. Integrating and evaluating neural word embeddings in information retrieval. In Laurence Anthony F. Park and Sarvnaz Karimi, editors, *Proceedings of the 20th Australasian Document Computing Symposium, ADCS 2015*, pages 12:1–12:8, 2015.